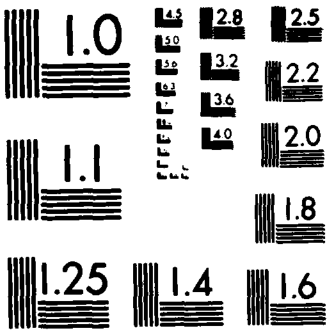
AD-A174 617 PROTOTYPE OF A COMPUTER METHOD FOR DESIGNING AND 1/2
ANALYZING HEATING VENTIL (U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF SYST S J BARLOW
UNCLASSIFIED SEP 86 AFIT/GEM/DET/86S-1 F/G 13/1 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

(2)

PROTOTYPE OF A COMPUTER METHOD FOR DESIGNING AND ANALYZING HEATING, VENTILATING AND AIR CONDITIONING PROPORTIONAL, ELECTRONIC CONTROL SYSTEMS

THESIS

Steven J. Barlow
Captain, USAF

AFIT/GEM/DET/86S-1

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

PROTOTYPE OF A COMPUTER METHOD FOR DESIGNING AND ANALYZING HEATING, VENTILATING AND AIR CONDITIONING PROPORTIONAL, ELECTRONIC CONTROL SYSTEMS

THESIS

Steven J. Barlow
Captain, USAF

AFIT/GEM/DET/86S-1

AFIT/GEM/DET/86S-1

PROTOTYPE OF A COMPUTER METHOD FOR DESIGNING AND ANALYZING HEATING, VENTILATING AND AIR CONDITIONING PROPORTIONAL, ELECTRONIC CONTROL SYSTEMS

THESIS

Presented to the Faculty of the School of Systems and Logistics
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Engineering Management

Steven J. Barlow, B.S., P.E.
Captain, USAF

September, 1986

Acknowledgments

A note of thanks and appreciation is extended to Major Steven T. Tom, my thesis advisor, and Captain Jay Santos, my thesis reader, for their guidance, suggestions and assistance. To say they were instrumental to this thesis is truly an understatement.

Thanks are also extended to Dr. Donald R. Clark of the National Bureau of Standards, Mr. Ed Wilson and Mr. Larry Struthers of HQ AFESC, Mr. Mark Provost of Barber Colman and Mr. Al Zajac of Johnson Controls for their assistance in identifying and discussing some of the reference material I used in conducting this research.

Finally, my special love and deepest appreciation goes to my wife, Colleen and my son, Nicholas, whose first sentence was "Daddy do homework". Colleen's understanding, patience and moral support during these past fifteen months served as the cornerstone of our family and enabled me to complete the program.

Table of Contents

List of Figures

Abstract

The Air Force needs a better method of designing new and retrofit heating, ventilating and air conditioning (HVAC) control system. Air Force engineers currently use manual design/predict/verify procedures taught at the Air Force Institute of Technology, School of Civil Engineering, HVAC Control Systems course. These existing manual procedures are iterative and time-consuming.

The objectives of this research were to: (1) Locate and, if necessary, modify an existing computer-based method for designing and analyzing HVAC control systems that is compatible with the HVAC Control Systems manual procedures, or (2) Develop a new computer-based method of designing and analyzing HVAC control systems that is compatible with the existing manual procedures. Five existing computer packages were investigated in accordance with the first objective: MODSIM (for modular simulation), HVACSIM (for HVAC simulation), TRNSYS (for transient system simulation), BLAST (for building load and system thermodynamics) and Elite Building Energy Analysis Program. None were found to be compatible or adaptable to the existing manual procedures, and consequently, a prototype of a new computer method was developed in accordance with the second research objective. The prototype method developed the architecture needed to meet the manual procedure compatibility requirement and modeled three electronic components: a sensor, controller, and hi signal selector.

The method incorporates four programs, written in BASIC, and copies of the programs, flowcharts, and sample runs are included. The method was developed to be easily expandable and recommendations for further development are given.

I. Introduction

General Issue

The Air Force needs a better method of designing new and retrofit control systems for heating, ventilating and air conditioning (HVAC) systems for its facilities. Prior to 1979, Air Force engineers or contracted Architect/ Engineering firm engineers usually specified the general type of control system using a performance specification. The construction contractor often subcontracted the control system installation to a controls manufacturer who designed and installed the control system.

This method of manufacturer-designed HVAC control systems presented several problems to the Air Force. A major problem was that the designs were proprietary. As a result, Air Force maintenance technicians often had difficulty understanding the documentation and servicing the hardware. When the HVAC control systems subsequently needed to be recalibrated and repaired, no one with the necessary knowledge of the control systems was available. Another problem was that the vendor-designed control systems were usually standard designs (standard, that is, for each particular vendor) and consequently, the control systems usually were not optimized for each specific HVAC system. Furthermore, the Air Force design engineers usually did not fully understand how the control systems operated due to the engineer's indirect role in the control system design. This fact made it difficult for them to review the design or to inspect the installation.

In their study, Schultz, Kenna and Kapka provided strong evidence that showed that the state of disrepair of HVAC control systems in Air Force facilities was alarming, primarily due to the overall lack of knowledge on both the engineering and technician level, Air Force wide (15).

Due to the need to combat this prevailing lack of knowledge, the Air Force Institute of Technology (AFIT) School of Civil Engineering developed a course entitled HVAC Control Systems. (This course was originally titled Facility Energy Systems). This course combats the problem by teaching Air Force engineers how to design and analyze HVAC control systems. The design procedures taught in this course insure the designs contain all information that the technicians need to maintain the control systems. The history and the content of material taught in HVAC Control Systems is described in detail in the background chapter of this thesis. However, at this time it is necessary to understand that Schultz and Kenna developed a new method for designing heating, ventilating and air conditioning control systems and the method has been constantly refined by their successors at the School of Civil Engineering.

Schultz and Kenna followed a "generic" approach, basing all designs on a few fundamental components which can be supplied by many manufacturers. Schultz and Kenna's approach has many advantages which are described in more detail in the literature review chapter of this thesis.

Research Problem

Despite its strengths, the Schultz and Kenna method is time consuming since, at the present time, all computations must be done by hand. Furthermore, if the designer optimizes the system to conserve energy (as he should), the procedure becomes iterative. Consequently, it is cumbersome to use the method as it currently exists. A designer must perform many hand computations to predict the control system responses and verify that the system will operate as intended under all anticipated operating and design conditions.

As a result of designers neglecting to perform this iterative, time consuming process, the control system designs are seldom optimized and may be inadvertently designed with inherent deficiencies. This fact leads to significant energy waste.

Research Objectives

The underlying goal of this research is to produce a computer method of designing HVAC control systems which the Air Force engineers can use to solve the problem identified: namely, to speed up the time consuming, iterative design/predict/verify process. To this end, two objectives were identified:

Research Objective Number 1. Locate and analyze an existing computer-based method and modify it if possible to meet this need.

Research Objective Number 2. Develop a new computer-based mehtod that is fully compatible with the existing manual design/analysis procedures taught at Air Force Institute of Technology, School of Civil Engineering. This objective was only intended to be pursued if an appropriate existing computer method could not be located or expediently modified.

As discussed in Chapter IV, an appropriate existing computer method was not found so the research concentrated on objective number 2.

Scope and Limitations

Research for this project was limited to the HVAC control systems. Other system components (such as air handlers, heating and cooling coils, dampers, valves, etc.) were only considered in the context that they were the objects of the HVAC control system.

Furthermore, only a specific HVAC control system type was studied: namely, a conventional electronic proportional HVAC control system as defined AFR 91-39 (4) (currently in draft form) and in Haines (9:Ch 1). However, from a systems modeling perspective, electronic proportional control systems and pneumatic proportional control systems are nearly identical. As a result, the computer method produced through this research can easily be adapted to model a pneumatic control system.

The reader who is familiar with the subject of this research will note that this research did not consider direct digital control (DDC) HVAC control systems or energy management and control systems (EMCS). However, these two specialized areas of the heating, ventilating and air conditioning controls industry are already receiving the emphasis of the controls industry's attention. In fact, the researcher observed that a very large amount of the current research in the HVAC controls area focuses on advanced computer-based control applications which are essentially nothing more than advanced direct digital control.

Although the HVAC controls industry labels direct digital control as the <u>accepted standard for new control systems</u> (19), current Air Force policy prohibits the use of direct digital control in AF facilities (5). A major reason for this ban is that both the hardware and the software used by these systems is proprietary and not easily serviced by Air Force technicians. If the industry eventually develops "standard" control algorithms for these devices and the Air Force allows their use, the computer method described in this thesis could be modified to model these algorithms. It should be noted that electronic and pneumatic proportional control systems studied under this research program comprise the vast majority of the existing HVAC control systems in AF facilities (15:19), (4). For the purposes of completness and for the benefit of the reader who is unfamiliar with this subject area, a short discussion comparing and contrasting conventional proportional control (the focus of this research), energy monitoring and control systems and direct digital control is presented in the literature review section of this thesis.

II. Background

This chapter will investigate two areas that will give the reader a better understanding of the need for this research. The first section, Amplification of Problem, describes Air Force problems with existing HVAC control systems and steps which have been taken to alleviate these problems. The second section, Investigation of Conventional and Computer-Based Control Systems, briefly compares and contrasts the two broad types of control systems implied in the title and reviews the current Air Force policy on each type.

Amplification of Problem

The following statement is taken from the executive summary of research accomplished by Schultz, Kenna and Kapka in 1982: "The typical HVAC system in the USAF is in a total state of disarray, and our technicians are ill-equipped to remedy the problem" (15:2).

Schultz, Kenna and Kapka based that statement on comments they received from the field while they were conducting an educational program on the operation and maintenance of HVAC systems. Consequently, they embarked on a research effort aimed at validating or refuting that statement. They used 18 volunteer Air Force HVAC technicians, (eight military and ten civilian), and several research methods including surveys, testing, laboratory training and observation to draw their conclusions. Their conclusions are summarized below (15:29-30):

a. Experienced Air Force HVAC maintenance personnel do not have the knowledge needed to effectively maintain typical Air Force HVAC control systems. but they posess the ability to learn to do so.

b. The typical Air Force HVAC system is not operating efficiently, nor is it being properly maintained.

c. Major improvements in the engineering support provided to the HVAC systems maintenance force are required if the Air Force is to realize the full energy savings potential that exists.

Furthermore, in a survey administered to the research subjects, the technicians indicated that (15:26-27):

a. Engineering assistance was sometimes available.

b. When received, the engineering assistance was <u>sometimes</u> helpful. (7 out of 18 agreed)

c. The engineering drawings available at base level did not contain the detailed information the technicians needed to properly calibrate and maintain the HVAC control systems and the technicians perceived the base level engineers could not produce drawings containing the required information.

Schultz, Kenna and Kapka provided additional evidence which suggested that Air Force engineers were unable to provide assistance to the technicians unless the engineers successfully completed AFIT's HVAC Control Systems (15:2). Thus, the purpose of this course is to teach Air Force engineers how to design HVAC control systems that are inherently energy conservative and maintainable. As part of this course, engineers learn to prepare design drawings which provide the technicians with the information they need to properly maintain the systems.

By developing the engineer's skills in designing such systems, HVAC Control Systems is directly addressing the lack of knowledge that exists at the engineering level. The course also indirectly addresses the problem of lack of knowledge at the technician level by forcing the engineers to include all appropriate maintenance information on the HVAC control system design drawings.

The concept of designing the control systems for maintainability means that HVAC control systems should be designed in a manner that facilitates maintenance rather than hinders maintenance. The control system should be logical and traceable. A logical design is one which is easy for an engineer (other than the designer) or a technician to understand. A traceable design is one in which the control signal can be followed from component to component, such that each component's effect on the signal is easily recognized. This design for maintainability concept is described more completely in Engineering Technical Letter 83-1. Some of the requirements established in this letter are:

a. Using remote sensors to accomodate locating controllers in the mechanical room.

b. Logically grouping controllers, adapters, relays and other standard control system components in easily accessible centralized controls cabinets.

c. Including pneumatic test ports and electronic-system terminal strips cross-referenced to the control schematic to facilitate troubleshooting and calibration.

In an article published in the ASHRAE Journal, Major Steven T. Tom presented the philosophy of the Engineering Technical Letter 83-1 design criteria to the HVAC industry (18:38-40). Tom provided an example where maintainability features such as those above were ignored. He states:

> It is difficult to check the calibration of a mixed air controller, for example, when the calibration data is not shown on the drawings and the controller itself is mounted on a duct 30 feet above the floor. [Tom:38]

Tom summarized the Engineering Technical Letter 83-1 mandated provisions of a HVAC control system designed for maintainability in his article. They are as follows:

a. A fully labeled control schematic which details all set points, throttling ranges, proportional bands etc.

b. A fully labeled electric ladder diagram.

c. A sequence of control which provides a brief written description of the control strategy.

d. A generic, functional description of each control component including all relevant data (specs) for each component.

e. A detail of the control panel which consolidates key control system components in a central, easily accessible location.

f. Legends and schedules which provide definitions of symbols used in the sequence of control description and clearly presents all data specified in the control schematic.

The advantages of using the Engineering Technical Letter 83-1 philosophy to design HVAC control systems become almost self-evident when compared to the performance specification approach described in the background chapter of this document. Whether one considers maintainability to be an advantage in its own right or as the vehicle used for achieving other more concrete advantages, maintainability is the goal of the Engineering Technical Letter 83-1 method.

Tom lists three advantages to designing for maintainability: accuracy, reliability and efficiency (18:39). In the end, it is clear that there are numerous advantages of designing heating, ventilating and air conditioning control systems in accordance with the Engineering Technical Letter 83-1 requirements.

Investigation of Conventional and Computer-Based Control Systems

As indicated previously, the focus of this research effort is to produce a computer method that will assist the controls system designer in analyzing a conventional proportional control system. This concept differs significantly from the concept of computer-based control systems; however, the difference between these two concepts could easily be misunderstood by the layman. In the method developed through this research effort, the computer is used as a tool in the design/analysis process; it is not physically part of the control system. However, in a computer-based control system, the computer is actually a "component" of the control system.

To understand the similarities and differences between conventional and computer-based control systems, it is first necessary to have a general understanding of what a control system is, what it's functions are, and how the control system serves it's function. A HVAC control system can be thought of as a group of interconnected components which control a heating, ventilating and air conditioning system. The control system components are connected in series and parallel arrangements, with each component performing a specific function within the overall system.

As defined in this thesis, four major elements are necessary for a HVAC control system: a sensor, a controller, a controlled device and a source of energy. Haines states that a control system consists of three necessary elements: a sensor, a controller and a controlled device (9:2). Instructional material published by Johnson Controls, Inc. also lists three necessary elements of a control system: a controller, a controlled device and a source of energy. The apparent difference between Haines and Johnson Controls is really only a matter of how the two different sources classify the essential elements. Haines considers the energy source as the means of accomplishing the HVAC system control (9:7) and the three necessary elements listed by Haines are the three different processes (with each process being accomplished by a specific type of control system component) that enable the energy source to control the HVAC system. The Johnson Controls viewpoint considers the energy source as one of the essential elements but it combines the sensing and controlling processes into one process (11:Ch XVI). Since the sensing and controlling processes can be considered as two distinct processes and it is clear that an energy source is needed, for the purposes of this thesis, a control system must contain the four necessary elements listed above as a minimum.

These four elements are combined to give the control actions needed to regulate the HVAC system. In addition, many auxilary devices, commonly referred to as adapters, are usually added to give the desired indication and control.

The first major element is the sensor. A sensor measures a variable condition (e.g. room temperature) and passes this "information" on to the second major element, the controller. A controller processes the information fed to it by the sensor and produces an output signal which eventually is received by the controlled device (e.g. a valve). Before reaching the controlled device, the controller's output signal may be "conditioned" by an adapter (e.g. a hi signal selector) as indicated above. The source of energy, the fourth element, is the medium of the control signal (e.g. air, electricity etc.).

Within a specific control system, two or more of the <u>broad</u> types of components (sensors, controllers, auxilary devices and controlled devices) could be grouped together to form subsystems, with each subsystem providing a specific function of the overall control system. For example, a temperature sensor could be mounted in each of two rooms and each sensor's output signal could serve as the input signal to a controller. The output signals of both controllers could then serve as the input signals to a hi signal selector. The hi signal selector output signal could then be passed on to a valve that regulates the supply of water to a cooling coil. The complete control system for these rooms would include heating controls, ventilation controls, fan controls, and various on/off and safety controls. However, the sensor, controller, and hi signal selector just described form a subsystem which controls the cooling coil based on the warmest room temperature.

While the above explanation and example of a subsystem of a HVAC control system is very simple, it illustrates the functions of individual components and specific subsystems of an overall control system.

The similarities between conventional and computer-based control systems are that both types contain the four basic elements listed above and both perform the same function; i.e., oversee the operation of the HVAC system. The differences between the two types lie in the characteristics of the physical equipment or components used to perform the functions provided by the four basic elements (and consequently, the method and accuracy of how the functions are accomplised). The distinguishing characteristics of each type of control system will now be investigated.

Conventional Proportional Control Systems. Conventional proportional control systems are usually further classified by their energy source, namely pneumatic (air), electric (line voltage alternating current or low voltage, i.e. 24 volts, electricity), or electronic (very low voltage, i.e. 15 volts or less, direct current electricity)

(Haines:7). Pneumatic and electric control represented the "state-of-the-art" for commercial and industrial sized HVAC systems from the inception of such systems through the late 1960s. Solid state electronic controls became popular in the early 1970s and represented the state-of-the-art until the introduction of computer-based control systems in the late 70's. Conventional proportional control systems make up a very large majority of the control systems installed in Air Force facilities today. For the purpose of this thesis, pneumatic, electric and electronic control systems are considered conventional systems.

A typical conventional control system an average commercial facility (such as a typical Air Force administrative building) might contain twenty to thirty components grouped into three or four subsystems.

A detailed explanation of conventional control theory is beyond the scope of this thesis. The key concepts to understand are that the controller is the "brain" of a conventional control system, the system is "built up" by linking together several "standard" components each providing a specific function, and the input-output equations for each component can be used to predict the control signal voltages or pressures from the sensor, to the controller, through the adapters and eventually to the controlled device.

Computer-Based Control Systems. Computer based heating, ventilating and air conditioning control systems can be broken down into two categories: Energy Management and Control Systems (EMCS) and Direct Digital Control (DDC) systems.

In an energy management and control system, a computer supervises the operation of the conventional proportional control system. Traditionally, (since the mid 1970's) the energy management and control system is added to an existing conventional control system which then becomes the slave of the energy management and control system.

Direct digital control can be considered the next generation of HVAC controls after Energy Management and Control Systems. The growth in micro computer applications during the late 1970s and early 1980s made these systems possible, and direct digital control became the state-of-the-art control technique during the same time period. Direct digital control uses digital microcomputers (instead of conventional proportional controllers) to perform controlling actions in HVAC control systems (12:47). In other words, the computer replaced the conventional controller as the brain of the control system. Thus, a direct digital control system differs from an energy management and control system in that with direct digital control, the computer actually replaces the conventional system's proportional controllers and adapters instead of just supervising their action (6:3).

Direct digital control offers several advantages over conventional proportional control systems. AFR 91-39 states that direct digital control has the potential of being the most versatile method of controlling heating, ventilating and air conditioning systems yet devised (5:par 1-6). Two additional sources list additional advantages of direct digital control as compared to conventional systems (12:81), (6). These advantages are summarized below:

a. Increased heating, ventilating and system efficiency which results in optimum energy (and cost) savings.

b. Lower capital cost (for buildings over 100,000 square feet).

c. Reduced maintenance requirements. (Eliminates need for periodic calibration and adjustment.)

d. Ability to test alternative control techniques quickly and at no additional cost.

e. Built in self-diagnostic cababilities.

f. Ability to be easily upgraded in future.

The disadvantages of direct digital control (as obtained from the same sources) include:

a. Limited availability.

b. Limited support.

Current Air Force Policy. The Air Force currently prohibits the use of direct digital control systems except for "experimental" cases which must be specifically approved on a case-by-case basis (5). However, with the above lists of advantages apparently far outweighing the disadvantages, one must ask why the Air Force maintains it's anti-direct digital control policy.

The answer lies in two broad areas: past Air Force experience with computer-based control systems and the issue of maintainability.

The reader should be warned that the above lists of advantages and disadvantages were developed from articles written by people in the direct digital control industry. Consequently, the lists may be considered biased. From the Air Force's perspective, the above list of disadvantages is not all inclusive, the disadvantages are more significant than they appear by just being listed and also, some of the above listed advantages are only claims made by direct digital control manufacturers. These claims have not yet been fully field tested because direct digital control systems have only started to become popular over the last five years or so.

The HVAC controls industry made similar claims about energy management and control system systems in the 1970's yet the AF has experienced a multitude of problems with it's initial energy management and control system endeavors. Most Air Force heating, ventilating and air conditioning experts agree that the majority of the energy management and control systems procured in the late 1970's are only now being brought to "operational" status. In fact, in May 1984, Major General Clifton D. Wright, Director of Engineering and Services, instituted a policy prohibiting any new energy management and control system purchases until the problems with the existing, non-operational energy management and control systems were solved. This energy management and control system policy was only rescinded in June 1985 (21). It seems logical that the "lessons learned" from the Air Force energy management and control system misfortune form the basis of the policy prohibiting direct digital control.

Both AFR 91-39 and Tom provide insight which supports the Air Force direct digital control policy. AFR 91-39 states:

> Because direct digital control is a new concept, new systems are being introduced almost every month and old ones are being withdrawn from the market. For this reason, direct digital control, except for "pilot projects" is currently prohibited by the Air Force Policy. [AFR:chp 1-6]

In his article, Tom addresses specific problems with maintaining direct digital control systems (one of the manufacturer purported advantages) from the Air Force's viewpoint:

> The design of maintainable control systems poses special problems for the Air Force, as we require "generic" designs and equipment. Current trends within the HVAC controls industry are toward Direct Digital Control and electronic "black boxes," many of

which are designed to be easily maintained by specially trained technicians. The term "specially trained" is crucial--the hardware and the maintenance procedures are specific to one manufacturer. This approach works very well when applied to a single building or a group of buildings with one manufacturer's control system, but it causes severe problems when applied to hundreds of control systems, each of which was purchased from the least cost bidder. [18:38-39]

Tom elaborates further on this issue by citing the frequent rotation of Air Force technicians which would create a need for continual retraining, especially in view of the potential for worldwide deployment. He contends that direct digital control systems are not more maintanable from an Air Force perspective.

III. Methodology

Four primary approaches were used concurrently to satisfy the two research objectives previously stated. They were: literature review, interviews with persons familiar with the various existing HVAC related computer packages and with the School of Civil Engineering mechanical faculty, independent study of the HVAC Control Systems course material and of an appropriate programing language, and finally, engineering development of a new computer method.

Literature Review

A review of all appropriate literature was conducted, including current refereed and trade journals, relevent Air Force design guidance and policy on HVAC control systems, and existing computer and/or mathematical models of HVAC systems and control systems. The literature review contributed significantly towards accomplishing research objective number 1.

The literature review also assisted in meeting research objective number 2 in that the researcher had to locate and study the appropriate material in order to determine whether to modify an existing computer method or develop a new one. However, this process would be more appropriately placed in the "independent study" approach and is discussed in that section.

Interview

The researcher conducted personal and telephone interviews as a means of gathering some of the information used in meeting both research objectives. The purpose of the interviews was to benefit from the expertise of some of the Air Force's and private sector's HVAC control system experts. The interviews were not intended to quantify underlying trends or predict phenomena; consequently, statistical analyses commonly used for such purposes was not appropriate for this research effort.

Interviews with the mechanical faculty of the Air Force Institute of Technology, School of Civil Engineering were conducted. These personal interviews served several purposes. The interviews were used to develop the criteria for the computer method which was used to design and analyze HVAC control systems in accordance with the objective of this research effort.

These interviews also aided in meeting research objective number 2 by providing information which augmented the information discovered through the literature review. Since the Air Force Institute of Technology, School of Civil Engineering mechanical faculty teach the manual HVAC control system design method, they provided invaluable insight for evaluating existing computer methods and for developing the structure and determining the input and output requirements of a new computer method.

The researcher also conducted telephone interviews with other Air Force and private sector HVAC control systems experts in conjunction with the literature review to initiate the search for existing computer design methods. Through interviews and the literature review, the researcher inspected five computer-based HVAC analysis packages which are discussed in more detail in the literature review chapter of this document.

Independent Study and Engineering Development

The third and fourth approaches that the researcher used were independent study and engineering development of the new computer method. These two approaches had to progress "hand-in-hand" as they were interrelated and could not be separated. Both approaches were aimed at developing the expertise needed to meet research objective numbers 1 or 2: namely, to modify an existing computer method, if appropriate, or to translate the existing manual HVAC control system design/analysis method into a new computerized method.

Regardless of whether the program selected was new or existing, the researcher had to build a foundation of knowledge in the appropriate programing language through self study of appropriate reference texts. Finally, the researcher had to refresh his knowledge of HVAC control system design by reviewing the course material for the AFIT HVAC Control Systems course.

To meet the requirements of research objective number 1, the researcher followed the steps listed below. These steps describe the actions needed to locate and adapt an existing computer program. A negative result at any one step precluded advancing to the next step. At this point, the particular program being investigated was considered not adaptable and the program was removed from further consideration. The steps used to evaluate an existing computer program were:

a. Locate the program and obtain a copy of any user's guides or other appropriate documentation.

b. Study the documentation to determine if the program focused on the HVAC control system (as opposed to the overall HVAC system) in sufficient detail to meet the research objective.

c. Identify the programming language and determine if the hardware requirements could be met by base level designers.

d. Compare the program inputs, outputs and structure to the desired program inputs, outputs and structure.

e. Determine whether or not changes to the existing program necessary to get the desired results were feasible.

f. Modify the existing program.

g. Run and test the modified program.

As described in the literature review section of this thesis, the researcher was not able to adapt any of the five existing computer programs investigated under this research effort. Thus, it was necessary to develop a new computerized design/analysis method in accordance with research objective number 2. The steps used to accomplish this task are listed below in chronological order.

a. Select and learn the programing language.

b. Review the HVAC Control System course material.

c. Determine the program inputs, outputs and structure.

d. Develop program architecture.

e. Select of the specific HVAC control system or subsystem to model.

f. Write, test, and document program.

g. Run program(s) to demonstrate the use of the new computerized design/analysis method.

IV. Literature Review

In accordance with the first research objective, the search through the current literature located several existing computer packages which simulate single HVAC system components independently as well as packages which simulate entire complex HVAC systems. This section describes experiments and simulations recently conducted in order to demonstrate the capabilities of three computer simulation packages, HVACSIM (which stands for HVAC Simulation), MODSIM (Modular Simulation), and TRNSYS (Transient System Simulation). All three were developed wihtin the last five years. Two additional computer simulation packages developed for HVAC applications exist, BLAST (Building Loads and System Thermodynamics) and the Building Energy Analysis Program developed by the Elite Software Company. Finally, a source which develops algorithms to be used in modelling HVAC systems and individual components was located and will be briefly described in this section.

Existing Computer Programs

The Center for Building Technology, National Bureau of Standards developed MODSIM in 1984 as part of a building system package, HVACSIM (10:752).

Clark et al. derived equations used to model several HVAC system components (pipes and ducts, a heating coil, and a three- way, mixing valve) and simulated these components using HVACSIM both individually and as a subsystem of a larger HVAC system (2:737). The components and subsystem were then set up in the lab and measurements were taken to verify the validity of the computer simulation. The predicted results from the MODSIM simulation matched the experimentally measured results very closely (2:745). Clark et al. also briefly discuss models for other HVAC system components and phenomena which are easily handled by HVACSIM including pumps, fans, flow splits and mergers, two-way valves, dampers, temperature sensors, and hysteresis effects.

Hill simulated a multizone air handling system serving two zones using MODSIM (10). Hill's objective was to demonstrate the capabilities of the MODSIM package, not to be an exhaustive study of a multizone air handling system or any particular control strategies (10:758).

He ran three simulations comparing the same morning start-up procedure using three different control strategies. The simulated system integrated models of the ducts, fans, three-way valve, hot water coil, mixing dampers, temperature sensors, controllers, and a room model. Essentially, Hill's three control strategies included various combinations of proportional plus integral control vs. only proportional control of the hot water coil and the zone mixing dampers. No experiments were conducted to verify the simulation validity. However, based on the combination of Hill's results (which analyzed the differences and similarities of the three strategies) and Clark et al.'s work (using HVACSIM) cited above, Hill's simulations were assumed to be valid (10:757).

As indicated above, Hill concluded that the systems were accurately simulated and meaningful results were obtained. He then listed some capabilities and features of the MODSIM package including:

a. Detailed simulation of large complex heating, ventilating and air conditioning systems and operational modes.

b. Accurate modeling of short-term dynamics with long-term system performance.

c. Çan investigate system dynamics and subsystem interactions.

d. Realistic treatment of pressures and flows.

e. Can produce modular simulations that allow rapid changes to the operational modes.

f. Easily solves differential equations using a state-of-the-art nonlinear equation solving package which accomodates accurate transient simulation. This feature makes MODSIM superior to systems only allowing steady-state simulation.

g. Modular organization of the program makes construction and alteration of simulations relatively easy (10:758).

Hill points out that "many ideas for the design of MODSIM were borrowed from TRNSYS" (10:752), which was developed prior to MODSIM. TRNSYS is a "component-based transient simulation program" (13:766) developed by the University of Wisconsin Solar Energy Laboratory in the 1981 time frame.

Lau et al. used TRNSYS to simulate the effects of four different control strategies on a chilled water plant for a 1.4 million square foot commercial facility controlled by a small energy management and control system (13:766). The strategies they simulated included determining optimum chiller, cooling tower, and pump strategies, chilled water storage mode regulation strategies, chilled water reset, and storage tank subcooling. Each control strategy was performed manually and they plotted the measured performance data against the simulated data to assure the TRNSYS simulations were accurate (13:776-779). Lau et al. developed component models for the chiller and cooling tower from the algorithms provided by Stoecker and the manufacturer's data for the equipment (13:772-773). They developed the models for the four control strategies simulated, but the article did not provide much detail on the actual modeling process.

Lau et al.'s overall objectives were to develop the models, determine if the control strategies could save energy and dollars, and demonstrate the use of TRNSYS. They concluded control strategies could be developed and the benefits of simulation with TRNSYS were clearly demonstrated. However, the authors made a statement (quoted below) which lends strong credence to the approach developed and advocated by the School of Civil Engineering mechanical faculty regarding achieving energy conservation through maintenance and maintenance through designing for maintainability.

> An examination of the AHUs [air handler units] showed that many of the outside air dampers were out of adjustment, causing excessive amounts of outside air during hot weather. The reduced cooling loads resulting from the proper adjustment likely result in as great or greater savings than all of the computer strategies. [13:773]

The final HVAC system computer simulation studied for this thesis was a TRNSYS simulation conducted by Hackner et al. (8).

Hackner et al. modeled an existing heating, ventilating and air conditioning system which consisted of five water chillers, four constant-volume and two variable air volume air handling units, and a two-celled cooling tower, all controlled by an energy management and control system. Their overall objective was to "determine operating strategies for heating, ventilating and air conditioning systems which incorporate system dynamics and interactions and potentially reduce energy use" (8:788). Their conclusions are summarized below (8:788):

a. Dynamic HVAC control strategies can be identified by computer simulation.

b. 3-4% savings result from operating under dynamic control (approximately 1 control decision per second) as opposed to making control decisions on an hourly basis.

c. Savings of 19% and 9% are possible (for the system studied) using optimal control strategies as compared to fixed set points and the current energy management and control.

d. Potential energy saving HVAC operating strategies were identified and reliable equipment models were developed.

As stated earlier, many (or possibly all) of the algorithms that TRNSYS uses for the HVAC system components were developed by Stoecker et al. (17). In developing these algorithms, Stoecker et al. intended to standardize the procedures and equations for representing HVAC system performance. Stoecker's algorithms consider only steady state response of the system and were developed using linear and multiple reqression techniques in conjunction with manufacturer catalog data. These equations can then be used for simulating practically every major HVAC system component and selected systems and subsystems (17:1-3).

Two additional computer-based methods of simulating heating, ventilating and air conditioning system operation are Building Loads Analysis and System Thermodynamics (BLAST) and the Elite Software Building Energy Analysis Program.

BLAST was developed by the US Army Corps of Engineers, Civil Engineering Research Laboratory during the 1970s and is somewhat expensive to run, requiring a mainframe computer. Although detailed study of this program has not been accomplished under this research effort, in a telephone interview Mr. Ed Wilson, HQ AFESC/DEMM indicated BLAST has provisions for limited analysis of varying control strategies (22).

The Elite Software Building Energy Analysis Program is a patented proprietary software package that analyzes entire HVAC systems. Like Blast, the Elite package focuses on the HVAC system, the building heating and cooling load computations, and the energy use characteristics of a given building with given occupants and building-use characteristics. The Elite package can be considered a mini, commercially available, version of BLAST designed to run on small micro computer systems.

Adaptability of Existing Programs

The existing computer analysis packages discussed above were not analyzed in detail. Instead, these packages were inspected to see if, in their present form, the packages were compatible with the manual procedures taught in the HVAC Control Systems course. The researcher found the packages were not compatible and then considered the possibility of modifying them so the modified computer packages could be used to augment the Air Force Institute of Technology manual method. However, it appeared that the possibility of modifying any of the existing computer programs would require more effort than developing a new program specifically for the intended purpose. Consequently, this possibility was ruled out by the research team (the researcher, the advisor and the reader) for the reasons indicated below.

In a telephone conversation with Dr. Donald R. Clark, one of the key personnel involved in the development of MODSIM, the researcher learned that running MODSIM required a minimum of a 32 bit mini computer with several megabyte storage capacity (20). These computer requirements far exceed the capability of the type of computers that the intended users will have access to. Furthermore, Dr. Clark advised that MODSIM was intended to be a "research tool" and the package uses sophisticated computer algorithmns to solve simultaneous differential equations. Dr. Clark suggested that attempting to "downgrade" MODSIM to suit the objectives of this research might prove to be a futile effort.

The research team also believed that attempting to modify TRNSYS would prove to be nonproductive. This assumption was made after reviewing the TRNSYS users manual, which indicated that, like MODSIM, TRNSYS models transient control system responses. Thus, the TRNSYS algorithms also solve or approximate simultaneous time-dependent equations. Base level design engineers typically have neither the computer resources nor the detailed input data needed to support this type of computer program, so modifying TRNSYS did not meet the objectives of this research.

In contrast to the overly-sophisticated control system analyses of MODSIM and TRNSYS (from the perspective of this research effort), BLAST does not consider the HVAC control systems in sufficient detail to be of any assistance to this research effort. Mr. Ed Wilson, AFESC/DEMM, Tyndall AFB, has approximately ten years experience in using BLAST and can therefore be considered a reliable BLAST authority. In a telephone conversation with the researcher, Mr. Wilson stated that BLAST is primarily used to estimate the overall energy consumption for specific building/HVAC system configurations. Although BLAST is capable of estimating the effects of altering control system parameters (in terms of building energy consumption), Mr. Wilson stated that BLAST does not calculate the various control system component input and output signals (22). Furthermore, BLAST requires a mainframe computer to run.

sensor, a single input controller and a high signal selector.

These three components are not sufficient to model even a simple control system, but they are sufficient to develop and test the basic computer programs needed to model any control system. This research effort concentrated on developing the basic modeling programs; the writing of additional subroutines to model additional components is a relatively straightfoward task and can be done at a later date. Nevertheless, due to the modular structure of the programs, the three components modeled can be grouped together to form typical subsystems which could be encountered in modeling an actual control system. Furthermore, all programs developed under this research effort can be run using the three components modeled with outputs obtained similar to those that would result from modeling an actual control system.

Structure of Computer Programs

Although the general criteria was well established early in the research effort, the overall structure of the programs was not established beforehand. Instead, the structure evolved through continued interaction among the research group. A discussion of the general underlying structure will first be presented and will be followed by a more detailed analysis of the data file manipulation

Consequently, modifying BLAST to satisfy the requirements of this research effort was not attempted.

In studying the user's manual for the Elite package, the researcher concluded that like BLAST, the Elite package does not model the HVAC control system in sufficient detail to be of major assistance in this research effort. Furthermore, since the Elite package is patented, the researcher felt that the company would not divulge computer algorithms they wrote in developing their package. For these reasons, attempting to adapt Elite Software's package to meet the objective of this thesis was not given further consideration.

Based on the above arguments, the research team elected to develop a new computer program which would be "custom-designed" to augment the existing manual procedures. The next chapter discusses the research team's efforts along these lines.

V. Development of New Computer Programs

As discussed in the previous chapters of this thesis, the Air Force Institute of Technology, School of Civil Engineering, Mechanical Faculty identified the need for this research. Both the researcher's advisor and reader are members of the mechanical faculty; the advisor is the mechanical department head while the reader is the member of the current five-man mechanical faculty with the most tenure. Thus, the advisor and the reader were very familiar with the problem at the onset of the research effort and they established the overall requirements for the computer method. Both the advisor and the reader were involved with this research effort throughout, and along with the researcher, the advisor and the reader acted as members of a research team. The relationship between the three research team members was similar to a consultant/client relationship with the researcher serving as a consultant and the other two team members serving as the clients.

Emphasis on Structure Development.

The research team soon realized that the first and foremost objective of the programing effort became to develop an overall programing structure that could be universally applied to any conventional HVAC control system and would permanently store the data. The focus of this research, therefore, was on developing a structure that can be applied universally and expanded in the future rather than writing a more detailed program that can only be used to model a few specific control systems. The difference between these two concepts will become apparent throughout the remainder of this chapter.

Criteria. Six criteria were established in developing the computer method in order to make the method useful as a teaching aid and eventually as a tool for Air Force HVAC control system design engineers.

The first criterion established by the advisor and the reader was that the user must be able to permanently store, retrieve, and use data on demand. If all the data which represents a particular control system had to be reentered for each subsequent run as required by an iterative optimization process, the computer method would be more cumbersome to use than the existing manual design/analysis procedures.

The second criterion established for the method was that it must be compatible with the existing manual method taught in the HVAC Control Systems course developed at the AFIT School of Civil Engineering. This method allows the designer to calculate what the input signal to each component and the output signal from each component will be for any given control system and set of operating conditions. As indicated in the previous chapters, the intent of the researcher's advisor and reader was to develop a computer program that would perform the same analysis but require less work. As discussed in previous chapters, none of the existing computer programs that were analyzed under this research effort focus on the control system in this fashion.

The third criterion used in developing this computer program was that the program be modular to accomodate future expansions and revisions. At the onset, the research team recognized that time might preclude the researcher from modeling all of the standard components which are commonly used in generic control system designs. Therefore, the goal was to structure the programing such that the program logic was complete in itself and would work reqardless of the number of components in a particular control system. In other words, in order to "analyze" a particular control system, each component of the control system would have to be individually modeled and then a master program would have to "oversee" the complete run. For instance, the master program would have to "know" how many and which components were used in the particular control system, "know" how the components were connected, select the appropriate computer model of each component in the required sequence, and then finally save and update all this information in a data file for this particular control system.

This goal was accomplished primarily through the use of five main programs which rely on subroutines for nearly all component-specific computer code. Accordingly, adding new components will only require writing new component-specific subroutines which can then be easily integrated into the programing structure developed under this research effort. This structure will be described later when the purpose and logic of each of the five programs is described in detail.

The fourth criterion was that the computerized method must be very flexible such that one standardized method would work for an infinite number of possible control component configurations. Next to being able to permanently store and retreive data, this was probably the most difficult criterion to satisfy. This criterion most clearly illustrates the difference between the two programming approaches discissed previously, namely developing a structure that could be universally applied versus writing a complete working program for one specific system.

The universal structure approach is needed because no two control systems are identical. Even if the same number of the same components were used in two similar control systems (an unlikely event), many variables could still differ between the two systems and thus require special treatment within a computer program. Such variables include the component programing instructions (the settings of the dials and gauges that a technician adjusts) and the component symbols as shown on the control system schematic drawing.

Another desired feature regarding the flexibility criterion is that the method developed through this research had to allow the user to use the symbol identifiers directly as they appear on the control system schematic. This insures that a computer printout of the modeled control system will directly correlate with the control system schematic, thus affording the user easy cross-referencing. Finally, the order of entering components into the computer is arbitrary. This provides flexibility by allowing the user to enter the components by component type, by subsystem or by any other means a particular user desires. Meeting this criterion also allows the user to add additional components during subsequent computer runs without re-entering all existing components.

The fifth criterion established for the computer program was that it be interactive. As a rule, the students who take HVAC Control Systems have little prior knowledge of HVAC control systems (as was discussed in background chapter). The course must therefore cover a lot of material in a relatively short time period.

Consequently, any additional effort required of the students to learn how to use the computer method would be unwelcome. By making the program interactive, any required computer training will be kept to an absolute minimum. Furthermore, the interactive requirement is important to the designer; if the computer program is too cumbersome, the designer will not use it.

The final criterion established for the computer method was that the computer programing language should allow maximum use by Air Force engineers.

The major factors used in choosing a language were who were the intended immediate and eventual users of the computer method, what computer equipment would most likely be avialable to them, and what language would the eventual users likely be most familiar with. It was decided the BASIC computer language best meet these requirements.

The intended users consisted of two groups. Initially, the users would be the faculty members currently teaching HVAC Control Systems at the AFIT School of Civil Enginering. However, the purpose of this research effort was to develop a computer method that would assist Air Force design engineers in the field and thus, the designers are the intended eventual users of the method.

Regarding the computer equipment available to both groups, the research team felt that the field engineers would have the most access to personal or micro computers.

Furthermore, the AFIT faculty currently use a micro computer system regularly. Finally, since BASIC is the language predominantly available to micro computer users for their own programing, the research team felt the users would be most familiar with BASIC.

It could be argued that since there are many different versions of BASIC currently used in the micro computer industry, any programs developed through this research effort might not run on the computer system that a specific user might have. The researchers concede this point. However, the researchers contend that any eventual user who has access to and routinely uses a specific micro computer system has the ability to make the minor syntax modifications that would be necessary to run the computer program on their systems.

The specific version used to develop the method for this research is SANYO BASIC, or more specifically, BASIC [MS-DOS] Ver 1.25 developed by the Microsoft Corp. The method was developed on a Sanyo MBC 550 series micro computer system. In addition, the computer programs developed have been entered in GW BASIC and successfully run on a WANG system at the AFIT School of Civil Enginering.

Components Modeled. In addition to developing the overall structure, the researcher modeled three of the most common electronic control system components: a temperature process used. Finally, the programing logic will be closely examined for each program of the computer method.

Overview of Structure. The structure of the computer approach can be best understood when both a functional context and an operational context are considered simultaneously.

Functional Context. The functional aspect of the structure evolved around the realization that there were essentially four tasks which needed to be accomplished in order for the method to produce the desired results. As a result, the method consists of four main programs, each providing one essential function.

First a data file had to be created which represented a snapshot of the control system schematic drawing in terms the computer could understand. The program written to accomplish this is CMODCRD (which stands for Control MODel CReat Data). At this point it is sufficient to understand that through CMODCRD, the user creates one permanent data file for each specific control system modeled. In BASIC jargon, the data file is called a random access data file and the data files are permanently saved on a floppy diskette until modified or removed by the user. In the course of running CMODCRD the user loads each control system component into the data file one-at-a-time and inputs all of the programing instructions and other known information (such as controller setpoint and the space temperature). These files will be referred to as data disk files.

Once a data disk file (representing a specific control system) has been created, the user must run an execution program, which is called CMODEXC (for Control MODel EXeCution). CMODEXC performs all calculations necessary to determine the input and output value for each component in the control system. The CMODEXC program also updates the data disk file with the results of all computations.

The third program in the CMOD series is CMODPRD (for Control MODel PRint Data). CMODPRD prints out the contents of the data disk file by component type. CMODPRD can be run at any time as long as a data disk file exists. The researcher recommends the user runs CMODPRD immediately after creating a data disk file to insure the data disk file created contains no errors that may have been inadvertantly entered when the file was created. Of course, the user must run CMODPRD after the execution program has been run to actually see the results of the computations, i.e., the output values for all control system components.

The fourth and final working program of the CMOD series is CMODEDD (for Control MODel EDit Data). CMODEDD allows the user to selectively change information stored in a data disk file. This enables the user to change specific parameters for any control system component or to modify the control system configuration. This is done in order to be able to predict the effect that any such change would have on individual control component outputs and on the overall control system performance as well. Thus, the user would use CMODEDD to change a specific component parameter, rerun the execution program, and rerun the print program. Then by comparing and contrasting the printouts of the edited data disk file to the original data disk file, the user can optimize a new control system design or can recommend improvements to an existing control system.

In addition to the four working programs just described, the CMOD series contains a fifth documentation program called CMODDOC (for Control MODel DOCumentation). CMODDOC briefly explains the purpose of the CMOD series of computer programs and gives the user basic instruction in running the programs. However, CMODDOC is not used in actually modeling a control system.

<u>Operational Context</u>. Although each of the four working CMOD programs serves a different function, they all use a similar three-tiered structure.

The first tier is the main program which contains all computer code that is not specific to any component type. The main program opens the specific data disk file requested by the user, orchestrates the transfer of program control among the appropriate component-specific subroutines, tracks and advances any counters and closes the data disk file (Refer to Chapter V, Structure of Computer Programs, Random Access Data Disk Files for explanations of opening and closing data files).

The second tier contains all the component-specific subroutines (used in CMODCRD, CMODEXC, and CMODEDD), or for-next loops (used in CMODPRD). Regardless of a particular program's function (i.e., create, execute, print or edit) any time a program manipulates the data from the data disk file for any component, it does so through a subroutine written specifically for that type of component. Thus, each of the four working programs contains one subroutine (or dedicated for-next loop) for each of the three component types currently modeled. Note, there may be additional subroutines nested within a component-specific subroutine. Furthermore, the program line numbers for each component type are the same within CMODCRD, CMODEXC, and CMODEDD. Specifically, the controller subroutines are found in the 1000 series of program line numbers, the temperature sensor subroutines are in the 2000 series of numbers and the hi signal selector subroutines are in the 3000 series numbers. Consequently, adding new components to the computer method would require writing three new subroutines (one each for CMODCRD, CMODEXC, and CMODEDD) and one for-next loop (for CMODPRD) for each new component type.

The third tier of the operational context refers to the method chosen for transferring data between the actual CMOD programs and the specific data disk file being used. The precise programing code is established in the programing language and must be followed. This procedure is described in detail in Chapter V, Structure of Computer Programs, Random Access Data Files. However, the research team had to decide how often data should be read into the program from the data disk file and written onto the data disk file from the program.

The research team considered two options. The first was to read the entire contents of the data disk file into the program one time, at the beginning of the program, and then write the new or updated data onto the data disk file one time at the end of the program. The second option was to read from the data disk file and write to the data disk file each and every time the computer addresses (creates, executes, edits or prints) a specific component in the control system.

Without an in-depth analysis, the first option appears to be more straightfoward and therefore would be the preferred option. However there are numerous obscure disadvantages. The first option requires some sort of array or subscript system be created in the main programs to correlate with the various (unknown) numbers of various (unknown) types of components. (Keep the universal application criterion in mind). Furthermore, in BASIC, subscripts and arrays must be numeric (e.g., TC(1), TC(2), TC(3), ...) not alpha (e.g., TC(A), TC(B), TC(C), ...) (7:89). This fact either precludes the use of symbols as they appear on the schematic drawing or necessitates some elaborate "behind the scenes" process of converting alpha symbols into numeric subscripted symbols. Basic data disk files can only be stored as alpha data so the computer would have to keep track of all the various subscripted symbols that are used in the subroutines and main programs and reconvert them to alpha symbols in order to re-store them in the data disk files (16:7-14).

Although a program could be written which would solve these problems, it would be extremly complicated and difficult to troubleshoot. Consequently, the researcher chose the second option, i.e., constantly transferring data between the data disk files and the programs. This process is not as cumbersome as it may sound and it lends itself to the subroutine structure.

Whether the function of the subroutine is to execute, edit or print, the procedure is the same. Specifically, the subroutine accepts control from the main program, goes to the data disk file, locates the specific component it is looking for, reads the specific piece of data it needs into the computer's memory, performs its function, writes the "new" value over the old value in the data disk file and finally, transfers control back to the main program. The procedure for the create subroutines is similar except that the "search" steps are obviously eliminated.

The second option process may sound cumbersome but it eliminates the need for arrays, subscripts, converting and reconverting symbols, and elaborate "bookkeeping". Therefore, all things considered, the researcher chose the second option and used the continuous data transfer method in all four working programs of the computerized method.

Random Access Data Disk Files. Random access data disk files form the foundation of the computer method. Mastering the use of these files proved to be the critical step in developing the four working CMOD programs. In fact, building the structure around the random access data disk file technique was the single most important factor that enabled the researcher to satisfy the three most important criteria, i.e., permanent storage of data, universal application and flexibility.

For completeness, this section will provide a brief overview of the two file-handling techniques (random access and sequential access) generally available in microcomputer BASICs. Then, the method of storing and retrieving data for random access data disk files will be examined. Finally, a map representing the contents of the data disk file that was created for an eleven component subsystem shown in figure 1 will be analyzed.

File Handling Techniques. In his book Secrets of

Better Basic, Ernest Mau states that BASIC offers the programmer a choice between two file-handling technique -- random access and sequential access (14:236). He claims that each technique has its advantages and disadvantages and the programmer should choose the technique that is most appropriate for a particular application. Mau provides brief yet clear explanations of these two data-handling techniques and his explanations are quoted below.

> As the name implies, a random-access file allows you to read data from or write data to a file at random. You can record or retrieve form any numbered "record" you desire without having to read through all the data prededing that record. This provides a fast method of retrieving individual items from large accumulations of data. Additionally, when a random file is "opened", you can read from one record, write to another record, read form another and so on. In short, you can mix storage and retrieval operations as you please.
>
> A sequential-access file may be used to read or write data only in consecutive order from beginning to end. When it is opened as in "input" file, data may be read but not written. When it is opened as an "output" file, data may be written but not read. [14:236]

Based on Mau's explanation above and the continuous data transfer method (the second option discussed in the previous section), the researcher felt that the random access technique was clearly the best choice for the computer method developed under this research effort.

Data Storage and Retrieval Method. The method of storing and retrieving data in a random access data disk file seems awkward at first, but it is really quite methodical and straightfoward. To fully understand the method, one must first generally understand what is going on "inside" the computer. Mau provides a very good explanation of microcomputer systems (Mau:Ch 1) and also of the process of working with random access data disk files (Mau:Ch 6). Detailed explanations such as Mau's are outside the scope of this thesis. The reader who desires a better explanation of the data storage and retrieval process than the simple explanation provided herein should consult the above reference.

In order to transfer data between a random access data disk file and the actual computer program, a single record of the random access data disk file is transferred as a unit of data to or from a random-file buffer (herinafter referred to as the buffer). Thus, when writing to a data disk file, data is transferred from the computer memory to the buffer to the data disk file (disk drive). The sequence of programing steps needed to write a record to the random access data disk file, along with the associated computer statement or function (in parentheses) is:

1) open data disk file	(OPEN)
2) field buffer	(FIELD)
3) convert numeric data to string &	(MKI$, MKS$ or MKD$)
write from memory to buffer	(LSET)
4) write from buffer to data disk file	(PUT)
5) close data disk file	(CLOSE)

In order to read the record back, the entire record must be transmitted to the buffer and the buffer must then be "decoded" by the program. The sequence of programing steps needed to read the record back, along with the associated computer statements or functions is:

1) open data disk file	(OPEN)
2) field buffer	(FIELD)
3) read from data disk file to buffer	(GET)
4) decode data &	(CVI, CVS or CVD)
read from buffer to memory	(equal sign)
5) close data disk file	(CLOSE)

Microsoft BASIC stores all data in string (character) form when using random access files. Consequently, numeric data has to be converted to strings for storage on the data disk file and converted back from string to numeric form upon retrieval. The process of converting from numeric to string is performed by the MKI$( ), MKS$( ), and MKD$( ) functions for integers, single precision real numbers and double precision real numbers respectively. (Single precision real numbers contain 7 or less significant digits, while double precision contain 8 or more). The process of converting the strings back to numeric values is performed by the CVI( ), CVS( ) and CVD( ) functions for integers, single precision and double precision numbers respectively (14:237). The MKx$( ) functions will always be used in conjunction with an LSET or RSET function (for left or right justify) when writing data onto the buffer from the memory. Note, there is no function opposite of LSET or RSET when reading from the buffer to memory.

The final nuance to observe in the data transfer process is that the variable names which represent the "same" piece of data (string or numeric) in both the data disk file and in the program cannot be the same. For exmple, if the variable name SP$ is used to represent a controller set point in the data disk file, then SP (the same variable name) cannot be used in the program. However, to establish a one-to-one relationship between program variable names and data disk file variable names, the researcher added the letter "I" (for interactive) to the end of all variable names in the program which represented the value for the same piece of data in the data disk file. Thus, SP$ represents the value of a controller set point in the data disk file while SPI represents the value of a controller set point in the program.

At this point an example of some actual computer code will be analyzed to demonstrate the use of the data storage and retrieval process.

Suppose a user wants to add a third record to an existing data disk file called "CONTROL". CONTROL currently contains two records. Each record contains only one user-furnished piece of data, the controller set point (SP$). The user wants to set the set point of the new record = 70.0. The computer code to add the third record would be:

```
10 OPEN "R", #1, "CONTROL",4
20 FIELD #1, 4 AS SP$
30 SPI = 70.0
40 LSET SP$ = MKS$(SPI)
```

```
50 PUT #1,3
60 CLOSE #1
```

Line 10 opens the data disk file with the filename of "CONTROL". It tells the computer which specific data disk file to be ready to use to store or retrieve data. The "R" tells the computer that CONTROL is a random access data disk file. The #1 tells the computer that CONTROL will hereinafter be referred to as file #1. The 4 tells the computer that length of the longest record in file #1 is 4 bytes long. (Integers require 2 bytes, single precision numbers require 4 bytes and double precision numbers require 8 bytes).

Line 20 fields the buffer area reserved for file #1. The buffer area can be thought of as a blank sheet of lined loose-leaf paper but with variable length and width. Each line would represent a record. The maximum number of records (paper length) permitted is 32767 while the maximum record length (paper width) is 256 bytes. Each record can be divided into as many groups of bytes (i.e. fields) of equal or different lengths as the user desires within the 256 byte constraint. Each field must be assigned a valid string variable name. The field statememt only divides the buffer into fields, assigns the field names, and determines the length of each field. Thus, the statement establishes the fields but it does not write anything into them. Any number of field statements (up to 32767) can be specified for any one data disk file. Thus, from line 20, there is only one field, 4 bytes long, named SP$. Since there is only one field, the record length equals the field length in this case.

Line 30 sets the program variable SPI equal to 70.0. Variable names are assumed to represent single precision numbers unless otherwise specified by attaching a suffix to the variable name (% = integer, ! = single precision, # = double precision, and $ = string).

Line 40 does two things. First it converts a numeric value into string form using the MKS$( ) function. Then, through the LSET statement, it stores or writes the string data onto the buffer by assigning the string data to the variable named SP$.

Line 50 contains the PUT statement that actually writes the data onto the data disk file (i.e., the disk drive) from the buffer. Note that the data was put in the third record of file #1.

Finally, the close statement in line 60 closes the file so that no more data can be transferred into or out of the data disk file. If the #1 was omitted, the close statement would close all data disk files that were open at that time. The maximum number of data disk files that can be opened simultaneously is 15. Closing a data disk file provides an element of safety since the data file could be ruined by an abnormal program termination such as a power interruption, system reset, disk drive malfunction and so on (14:243).

The computer code needed to retrieve the data just entered uses identical OPEN, FIELD and CLOSE statements but the inverse of the PUT statement and the MKS$( ) function. The computer code to retrieve the third record would be:

```
10 OPEN "R", #1, "CONTROL", 4
20 FIELD #1, 4 AS SP$
30 GET #1,3
40 SPI = CVS(SP$)
50 PRINT SPI
60 CLOSE #1
```

The GET statement in line 30 reads the third record of the data file CONTROL onto the buffer from the data disk file (disk drive).

Line 40 converts the string data in the buffer to numeric data and the numeric variable (SPI) is set to the single precision numeric value in the program. Thus the value was converted then read from the buffer into the memory.

When line 50 is executed, the value of 70.0 will be printed out.

Data Disk File of Subsystem Analyzed. Figure 1 is a schematic drawing of an eleven component subsystem that was analyzed using the CMOD computer programs developed under this research effort. This is not a typical subsystem that would be encountered in an actual control system in the field. However, it uses all three of the components modeled and contains a sufficient number of components to

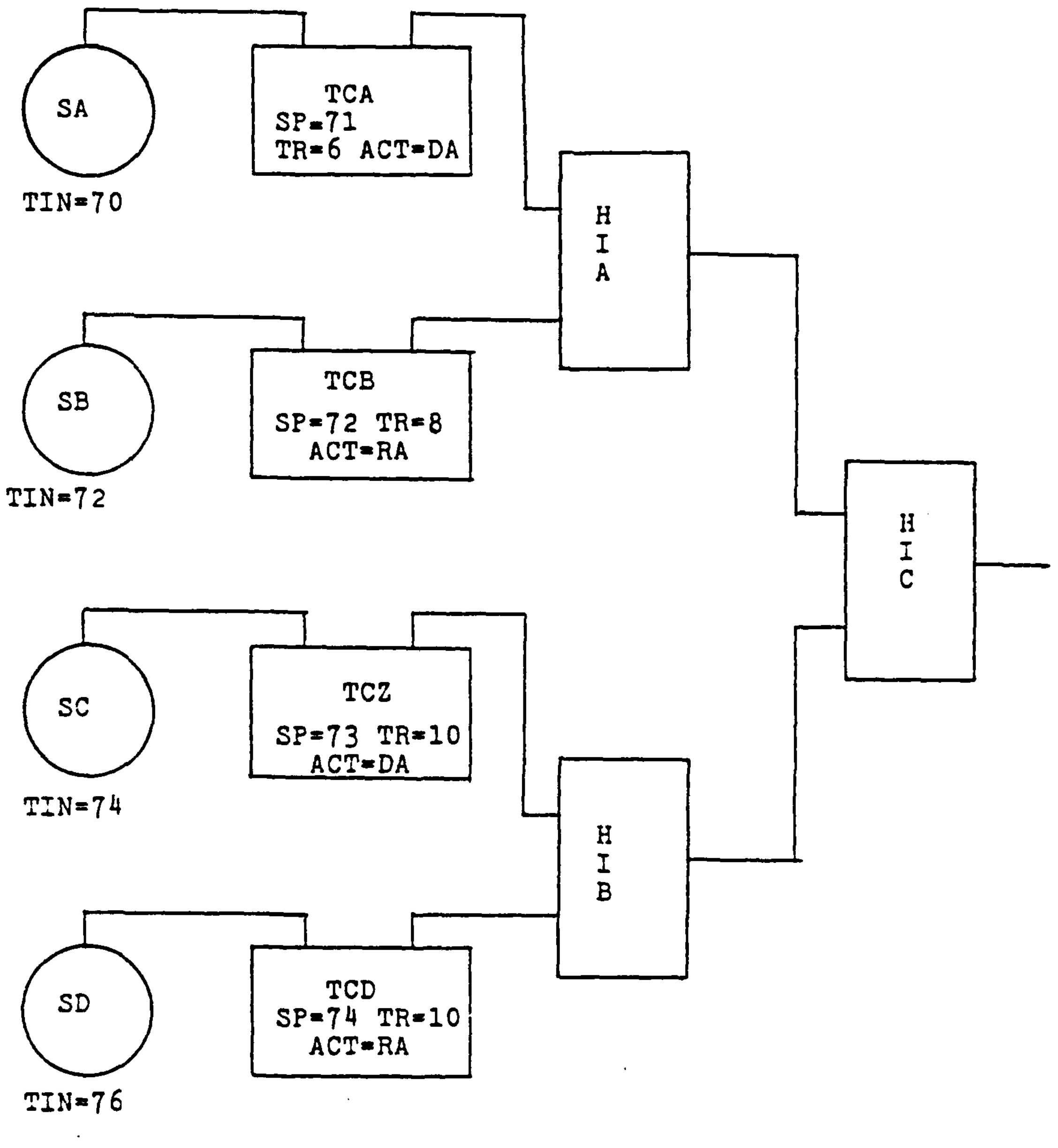


Figure 1. Schematic of Subsystem Modeled

provide enough complexity to sufficiently test the CMOD programs and demonstrate the utility of the method.

The schematic drawing shown in figure 1 furnishes a trained control system engineer or technician with all the information needed to analyze the control subsystem represented. However, the information contained on the schematic drawing must be "translated" into computer data in order for the computer to analyze the control system. The random access data disk file created specifically for the subsystem shown in figure 1 provides this translation. Thus, the data disk file furnishes the computer with all the information the computer needs to analyze the control system.

Description of Subsystem. The subsystem shown in the schematic contains four temperature sensors (SA, SB, SC, and SD), four single-input controllers (TCA, TCB, TCZ, and TCD) and three hi-signal selectors (HIA, HIB, and HIC). Each of the four controllers will produce an output voltage based on the controllers preset programing instructions (set point, throttling range and action) and the input temperature the controller is "reading" from its respective temperature sensor. The output voltages from TCA and TCB serve as the two input signals to HIA while the output voltages from TCZ and TCD serve as the two input signals to HIB. Both HIA and HIB select the higher of their respective two input signals and pass the higher signal on as their respective output signals. Thus, the outputs of HIA and HIB serve as the two input signals to HIC. Finally, HIC selects the higher of its two input signals and passes the higher signal on as it output signal. The output signal of HIC is the output value of the overall subsystem. The computer printout generated from modeling this subsystem using the CMOD method is included as Appendix C.

Representation of Data Disk File. Figure 2 is a representation of the random access data disk file for the subsystem shown in figure 1. The information stored in the computer or on the floppy disk is not actually stored in tabular form as shown in figure 2, however, the information can be best understood by a human if it is shown in tabular form. Figure 2 represents the data disk file as it exists immediately after being created by CMODCRD but before being executed and updated by CMODEXC.

Several key features of the data disk file representation will now be explained. First note that each "record" actually consists of two lines. The first line of each record results from the FIELD statement as discussed in the previous section. Specifically, the record is divided into fields of length specified in the statement and the fields are given field names of string form. In short, the FIELD statement reserved blocks where future data will be stored. The second line of each record

RN$	CMPC$
12	11

N$	TYP$	SY$	OTPT$	SP$	TR$	ACT$	INSY$	TIN$
2	1	TCA**	999	71	6	DA	SA***	70

N$	TYP$	SY$	OTPT$	STIN$
3	2	SA***	70	70

N$	TYP$	SY$	OTPT$	STIN$
4	2	SB***	72	72

N$	TYP$	SY$	OTPT$	INY1$	HIN1$	INY2$	HIN2$
5	3	HIA**	999	TCA**	999	TCB**	999

N$	TYP$	SY$	OTPT$	INY1$	HIN1$	INY2$	HIN2$
6	3	HIC**	999	HIA**	999	HIB**	999

N$	TYP$	SY$	OTPT$	SP$	TR$	ACT$	INSY$	TIN$
7	1	TCB**	999	72	8	RA	SB***	72

N$	TYP$	SY$	OTPT$	SP$	TR$	ACT$	INSY$	TIN$
8	1	TCZ**	999	73	10	DA	SC***	74

N$	TYP$	SY$	OTPT$	SP$	TR$	ACT$	INSY$	TIN$
9	1	TCD**	999	74	10	RA	SD***	76

N$	TYP$	SY$	OTPT$	STIN$
10	2	SD***	76	76

N$	TYP$	SY$	OTPT$	STIN$
11	2	SC***	74	74

N$	TYP$	SY$	OTPT$	INY1$	HIN1$	INY2$	HIN2$
12	3	HIB**	999	TCZ**	999	TCD**	999

Figure 2. Representation of Random Access Data Disk File

reflects the actual data that was written onto the record via the LSET and PUT statements. Recall however, that any numeric values stored in the data disk file are actually converted and stored in their string form in the data disk file.

Careful observation of figure 2 will reveal that the data disk file contains twelve records but there are actually only four different record configurations in this data disk file. Each record configuration was custom designed for either the record and component counters or one of the three components that were modeled under this research effort. Note however, that the first four fields for each component-specific record configuration (i.e., records 2 through 12) are identical.

The first record of every data disk file created in the CMOD system contains two fields, RN$ and CMPC$. RN$ stores the number of records the data disk file contains at any point in time. CMPC$ contains the number of components the data disk file contains. Since there is one record per component and the first record always contains the counters, CMPC$ will always be equal to RN$ - 1. These counters must be tracked in order to be able to tell the computer how many records it must search through while searching for specific data in the data disk file.

The four fields common to all component-specific records will now be described.

The N$ represents the record number for each specific field. Although the computer tracks record numbers internally, the user can not "see" the computer's internal tracking numbers. However, the programing logic required for the search routines in the CMOD programs required that the record numbers be "visible" and these numbers be used to locate specific pieces of data. Consequently, the computer's internal tracking numbers were duplicated in the data disk file.

The TYP$ represents the particular component's type. This number is used to select the appropriate subroutine in the CMOD programs based on the component type. Currently, single input controllers are type 1, temperature sensors are type 2, and hi signal selectors are type 3. As new components are added, the type numbers should continue with type 4 and so on.

The SY$ represents the specific component's symbol as taken from the control system schematic drawing. In establishing the field length of SY$, the researcher allowed five character spaces for any symbol. However, the schematic drawing symbols can vary from one to as many as five symbols. This presents a problem when "searching" for specific component symbols in the edit program because the computer recognizes blank spaces as characters. For example, if the user forgets to insert two blanks after entering the three-character long symbol TCA, the computer will not "find" TCA because it will be looking for "T C A blank blank". This problem was overcome by instructing the user to add asterisks (*) to the end of all symbols entered such that all character symbols will be five characters long. The user could opt to not add *s and use blanks where appropriate. However, the user must be extremely careful to consistently use the blanks throughout all four CMOD programs.

OTPT$ is the final field name common to all records. OTPT$ represents the output signal of the specific component modeled on each specific record. Note that the data file creation program (CMODCRD) inserts 999 as the output for all components except temperature sensors when the component is first modeled. Once the execution program (CMODEXC) is run, the value for OTPT$ is updated.

Record 2 represents a single input controller. Records 3 and 4 represent temperature sensors. Record 5 represents a hi signal selector. The order in which these records are listed is entirely arbitrary. The order in which the user entered the components while running CMODCRD determines the order of records in the final data disk file. The same system could therefore be represented by numerous data disk files depending on the order in which the individual components were entered.

The remaining five fields of the "controller" record will now be explained. SP$, TR$ and ACT$ represent the programing instructions for the controller. These symbols represent the controller set point, throttling range and action respectively. The next field, INSY$, stores the symbol of the component from which the controller receives it input signal. The final field, TIN$, records the controller's input temperature, i.e., the output signal of the temperature sensor identified in INSY$.

For the temperature sensor record, the only remaining field is STIN$. STIN$ stores the value of the temperature being read by the temperature sensor. In the mathematical model, the value is passed directly to the controller. Consequently, no calculation is needed to determine the sensor output as the input equals the output. For this reason, the output value is entered in the sensor record for OTPT$ when the user creates the sensor record under CMODCRD.

Finally, the remaining fields of the hi signal selector record are as follows. INY1$ and INY2$ store the symbols of the two components which provide the input signals to the hi signal selector. HIN1$ and HIN2$ store the values of the two input signals to the hi selector, i.e., the output values from INY1$ and INY2$ respectively.

At this point the relationship between the schematic drawing and the random access data disk file should be clear. The data disk file duplicates the schematic but in terms the computer can understand. When looking at a schematic, an engineer or technician can determine how the control system components are connected by following the lines which connect the components on the shcematic drawing. He also knows that the output value of one component is the input value to another component.

The computer knows how the components are connected through the disk file fields which store the symbols of the input components. Furthermore, the computer knows what the actual output values and corresponding input values are through the appropriate data disk file input value and output value fields.

However, there is still one critical ingredient needed to analyze a control system. The "analyzer" must determine the overall order in which the calculations must be made or he will not have all information needed to make all calculations necessary. For instance, in the system shown in figure 1, the designer could not calculate what the output value of HIA would be unless he first knew the output values of both TCA and TCB. This "order-of-calculation" requirement is intuitive and poses no real problem to the designer. However, the computer has no intuition and this requirement poses a significant problem, especially in the context of the universal application criterion. It is not possible to establish a standard order-of-calculation rule for the purposes of a computer method, since the very nature of designing control systems mandates that the designer be able to choose which type of component is used to provide an input signal to another type of component.

After carefully considering the order-of-calculation problem a "sophisticated" logic-based solution was not obvious. Consequently, the researcher relied on an exhaustive numerical procedure suggested by the researcher's advisor. This procedure will be analyzed when the computer code for each of the four CMOD programs is discussed in the next section.

Analysis of Computer Code Logic

The actual computer code will now be reviewed for each of the four CMOD working programs along with the associated logic. Each program is fully flow charted to assist the reader in following this discussion. The actual computer code is included in this document as Appendix A while the flow charts are included as Appendix B.

The Data Creation Program, CMODCRD.

This program serves two purposes. First it is used to create a new random access data disk file. CMODCRD is also used to add new components to an existing data disk file.

Lines 120 through 170 set some of the phrases that are frequently repeated throughout the interactive part of this program equal to short string variables. This speeds up the overall execution of the program and shortens the time required to type in the computer code. The GOTO statement in line 180 bypasses lines 190 - 230. These lines are only used the first time a particular data disk file is established to initially set the record counter (RN$) equal to 1 and the component counter (CMPC$) equal to 0. Thus, lines 190 - 240 create the first record of every new data disk file which contains the two counters. These counters are used as the upper limit in the numerous FOR/NEXT loops that are used throughout all four CMOD programs.

Lines 240 - 290 print out a short statement for the user's convenience which verifies which program the user is running. This feature is especially nice when the user is reviewing the printouts of the computer runs for a typical "modeling session" which may be over 20 pages long.

Lines 300 - 340 ask the user to type in the filename of the data disk file for the particular control system being modeled and then give the user a chance to correct any typos.

In lines 350 - 380 the user tells the computer whether the data disk file he specified is for a new or an existing system. If the user is creating a new data disk file, then program control is passed back up to line 190 in order to establish the record and component counter records as discussed above.

At this point (line 400) the computer prints out a menu with instructions and prompts the user to select the type of component he wishes to enter. After the user types in the appropriate number line 450 passes control to the appropriate component-specific subroutine. Since all three subroutines use essentially the same logic, only the single input controller subroutine will be analyzed.

Upon entering the subroutine, the first step accomplished is printing out the subroutine verification. Next, the data disk file is reopened (line 1020) and the buffer is fielded for both the counter record (line 1030) and the controller (line 1040). Line 1050 reads the counter record into the buffer from the disk file and line 1060 sets the current record and component counters in the computer memory.

The computer then tells the user the number of components currently stored in the data disk file and asks the user if he wants enter another controller (lines 1070 - 1110). Lines 1090 - 1110 give the user a chance to exit the controller subroutine if he mistakingly typed in the wrong number from the menu or it allows the user to enter another controller if the user elects to enter all of the same type of components at the same time.

If the user typed in "NO", the data disk file is closed and control is passed back to the main program. If the user typed in "YES" the computer increments both counters by 1 (line 1120) and then proceeds to prompt the user to furnish the input data as described in the previous section.

After the user types in all requested information, the computer repeats it all and asks the user to verify the input data for the controller is correct (lines 1210 - 1250). If the user indicated the data was not correct the computer prompts the user to retype in the correct data. If the user indicated the data was correct, the computer then updates both the counter record and the controller record using the LSET and PUT statements as described in the previous section (lines 1300 - 1350).

The GOTO statement in line 1360 then directs control back to line 1070 which once again, gives the user the option of entering another controller or exiting the controller subroutine and reentering the main program at line 480. Lines 480 - 490 ask the user if he wants to enter another component then transfers control back to line 400 on a affirmative response or to line 500 on a negative response.

Lines 501 - 505 print out a confirmation that the Create program is complete and prompt the user to select another CMOD program.

The Data Print Program, CMODPRD. Unlike the other three working CMOD programs, CMODPRD employs FOR/NEXT loops instead of subroutines for all of its component-specific computer code. The reason for using FOR/NEXT loops lies in a combination of the desired format of the printouts and the fact that "forcing" the use of subroutines would have only required extra lines of computer code but would have produced no tangible benefits.

It is desirable to have all components of the same type printed out together rather than have the components printed back in the order in which they were entered into the data disk file. The logic required to produce this like-component format mandates inspecting the entire data disk file and only printing out a record when the desired component type is found. FOR/NEXT loops are very well suited for this task. For example, in the three-component subsystem shown in figure 1, three passes through the data disk file were made using a FOR/NEXT loop for each pass. All component-specific records (records 2 - 12) were inspected on each pass. On the first pass, only the controllers were printed out. On the second pass, only the temperature sensors were printed out and it follows that on the third pass, only the hi signal selectors were printed out.

The FOR/NEXT loops indicated above could have been embedded in subroutines in order to make the print program fit the structure of the other three CMOD programs identically. However, the researcher felt the extra programing steps required would have slightly lengthened the execution time of the print program. Consequently, the subroutine structure was not used in this case. Nevertheless, the logic used in the print program is rather straightfoward. Lines 100 - 220 print out the program verification and prompt for the user to furnish the data disk file name.

Lines 240 - 280 open the data disk file and field buffer for all four record configurations. Line 380 gets the counter record to use the total number of records as the upper limit for the FOR/NEXT loop search.

All three FOR/NEXT loops use the same procedure. The controller loop resides in lines 390 - 440, the temperature sensor loop is contained in lines 500 - 550, and lines 620 - 670 make up the hi selector loop. Within each loop, the counter variable, C, varies from C = 2 to C = the last record number. Then the C'th record of the data disk file is read using a GET statement. If the component type indicated in the TYP$ field of the C'th record matches the particular value sought for in the particular loop, the contents of that record is printed. This procedure is repeated by incrementing C by 1 until all records (excluding the first, counter record) of the data disk file are searched one time per FOR/NEXT loop.

Each loop is preceded by the statements to print out the headers for the specific type of component (i.e., lines 310 -360 for the controllers, lines 460 - 490 for the temperature sensors, and lines 570 - 600 for the hi signal selectors).

<u>The Execution Program, CMODEXC</u>. As indicated earlier

in this chapter, the logic employed in CMODEXC relies on an iterative "brute force" approach. This approach makes the order in which the component outputs are calculated irrelevent because eventually, all the required input values will be computed if the calculations are repeated often enough. Consequently, the program was developed such that the program iterates as many times as there are components in the control system being modeled and each component's output value is computed one time per iteration. For example, in a 10-component system, 10 iterations will be made and each component will have its output value computed one time per iteration. In all, 100 calculations will be made. This procedure is needed in order to insure that by the last iteration, all components that pass on their output values as input values to other components have their "correct" output value stored in the component's OTPT$ field. Nevertheless, CMODEXC runs rather quickly (approximately 2 - 3 seconds per component) since the program does not consider transient responses and high-level mathematics are not used.

CMODEXC follows the same initial steps as the print program. Specifically, it prints out the program verification (lines 100 - 140), prompts for and verifies the data disk file name (lines 160 - 240), opens the data disk file and fields all records (lines 250 - 290), reads the counter record data from the disk file and sets the last record number and component count into memory (lines 300 - 320).

At this point, an iteration counter (ITER) is set equal to 0 (line 320) and a FOR/NEXT loop using the variable, I, and an upper limit equal to the number of records in the data disk file is used to start the iteration process (line 330). The computer reads the I'th record from the data disk file (line 340), sets the component type value from the record in memory (line 350), and selects the appropriate component subroutine based on the type value (line 360).

At this point, control is transferred to the appropriate component subroutine where the component's output value is computed and then stored in the output field in the data disk file record. Each component-specific subroutine computes the component's output value based on a mathematical model for that component. The equations used in the subroutines were taken directly from the manual procedure taught at the HVAC Control Systems course at the AFIT School of Civil Engineering. The computer code for each subroutine will be analyzed after the discussion of the main program logic is completed.

Control is transferred back into the FOR/NEXT loop in the main program and the counter, I, is incremented by 1 (line 370). The next record is read from the data disk file (line 340, again), the new component type is reset in memory (line 350) and the appropriate computation subroutine is once again selected and executed and the output value for that component is updated on the data disk file from within the subroutine. This process continues until the output values for all components in the control system have been calculated one time, thus completing the first iteration.

Line 380 then increments the iteration counter, ITER, by 1 and a short message is printed out informing the user that the ITER'th iteration is complete. The value of ITER is compared to the number of components in the system (line 410) and if ITER is less than the number of components, control is passed back to the FOR/NEXT loop for the next iteration. This iterative process continues until such time as ITER equals the number of components in the system. Thus, for an N component system, each component will have it's output computed N times and the logic insures that all components will have their "correct" output values stored in their respective output fields of the data disk file. The data disk file is then closed in line 460 and a program complete message is printed out in lines 480 - 530.

Only two components are modeled in the execution program subroutines; a single input controller and a hi signal selector. In the HVAC Control System equations, the temperature sensor simply passes the temperature it senses as its output value and consequently, no subroutine is needed.

Recall that before calling a particular subroutine, the component record has already been read into the buffer and the value of the component's type field has been set in the computer memory. Thus the first step in both subroutines is setting the appropriate input data in the computer memory. This task is accomplished via line 1010 in the controller subroutine and in line 3010 in the hi selector subroutine.

The remainder of the controller subroutine will now be analyzed. Lines 1020 - 1080 contain a FOR/NEXT loop which tells the computer to search through the data disk file until the record containing the controller's input component is located. Specifically, line 1020 varies the counter for this loop (J) between 2 and the last record in the file (K). Line 1040 reads the Jth record into the buffer and the IF statement in line 1050 checks to see if the symbol of the specific component being "searched" is in fact the symbol of the controller's input component. The IF statement in line 1030 is used to prevent the computer from re-reading the same record that the computer read prior to entering the subroutine.

If the symbols do not match in line 1050, J is incremented by 1 and the data disk file is searched record-by-record until the input component is found. Once the input component is "found", control exits the loop and line

1060 sets the input component's output value equal to T in the computer memory. Recalling that the controller's set point, throttling range and action were read into memory upon entering the subroutine, line 1090 directs control to either of two equations (lines 1100 or 1120) based on the controller's action. At this point all information required to compute the controller's output value is known and the equations assign the variable V equal to the computed output.

It is now necessary to read the controller's record back into the buffer from the data disk file (line 1140) because the information stored in that record was "lost" during the search for the input component. The final steps of the controller subroutine involve writing both the controller's output value and the value of the input component into the appropriate fields of the controller record. Lines 1150 and 1160 accomplish this task.

The hi signal selector subroutine uses the same general logic as the controller subroutine and therefore, the hi selector subroutine will be only briefly described. A FOR/NEXT loop is used to search the data disk file for the input components in order to obtain the value of the input components. These values are then used in the component equation which really does nothing more than select the greater of the two values. Both the two input values and the output value are updated in the hi selector data disk file record in the same manner as in the controller subroutine.

The Data Edit Program, CMODEDD. The data edit program allows the user to change the contents of any field in any record in the data disk file. The logic behind this program is rather straightfoward. After printing out the program verification, lines 140 - 270 print out a menu and prompt the user to select one of three choices; editing, adding components to, or removing components from an existing data disk file.

Although the menu appears to give the user three choices, the only task that the user can appomplish through CMODEDD is to edit an existing data disk file. Nevertheless, the three-choice menu is included in CMODEDD for the user's convenience. If the user attempts to add a new component to an existing data disk file under CMODEDD (line 290), the IF statement in line 300 passes control to lines 630 - 650 which ask the user to run the data creation program (CMODCRD) and CMODEDD is aborted. If the user attempts to remove a component from an existing data disk file, the IF statement in line 305 passes control to lines 660 - 710 which tell the user that he must create a new data disk file while omitting the undesired components from the new file. This method of "removing" components by creating a new data disk file is necessary because the researcher has not been able to locate or develop a method of removing individual records from a random access data disk file.

Assuming the user wants to edit an existing component, the computer follows the standard procedure of asking for the data disk file name and verifying (lines 310 - 380). In lines 400 - 440, the file is opened and the buffer is fielded. After reminding the user to use 5-character symbols, the computer prompts the user to enter the symbol of the component the user wishes to edit (lines 460 -480). Lines 490 - 500 read the counter record into the buffer and set the variable K equal to the number of records in the data disk file.

The FOR/NEXT loop in lines 510 - 550 is used to search through the data disk file, find the appropriate record, and set the value stored in the type field equal to the variable TYN. Program control is then passed on to the appropriate component-specific subroutine based on the value of TYN (line 560).

All three subroutines use the same logic so only the controller subroutine will be examined. Lines 1001 - 1010 print out the subroutine verification and then the values stored in the data disk file fields are read into the computer memory in lines 1020 - 1030. Lines 1040 - 1110 prints out all data currently stored in the data disk file for the controller being edited and then the computer prompts the user to select the specific piece of data that he wishes to change from a menu (lines 1140 - 1370). Note that in line 1320, the computer instructs the user that he must "edit" the input component (rather than the controller) in order to change the value of the input component.

The user then selects one of the four possible choices (indicated below) and the program control is passed on to the appropriate subroutine written specifically for each of the four data fields. These subroutines are contained in lines 1400 - 1430, 1440 - 1470, 1480 - 1510, and 1520 - 1550 for the set point, throttling range, action and input symbol respectively. Within each subroutine, the computer prompts for the new value, writes the new value to the buffer using an LSET statement, and then writes the new value to the data disk file using a PUT statement.

Upon exiting either of the three component subroutines, the user is given the option to edit another component (lines 570 -600). If the user answers affirmatively, control is passed back to line 460 and the processs is repeated. If not, the data disk file is closed (line 610), the "add" and "remove" instructions are bypassed (line 620) and the run-complete message is printed (line 730 - 790).

VI. Conclusions and Recommendations

Significant progress has been made toward meeting the overall objective of this research. This objective was to "computerize" the manual process of designing and analyzing HVAC control systems and thus speed up the cumbersome design/analysis process. As a result of this research, a prototype of a computerized method to assist Air Force engineers design and analyze proportional electronic HVAC control systems has been developed.

The method developed contains four working programs and a documentation program. In order to demonstrate the utility of the method and the operation of the four working programs, an eleven-component control subsystem (built up from three different component types) was analyzed using the prototype. A sample run of this analysis appears as Appendix C.

The remainder of this chapter will focus on the practical implications of the prototype computer method developed through this research effort and the researcher's recommendations for future wo~~r~~k needed to expand the prototype so that it is fully capable of analyzing any new or existing control system of the type specified above.

Practical Implications of Results

The objective of this research effort dictated that any progress made would be of practical rather than theoretical benefit. Although the work was based upon teaching methods used at the AFIT School of Civil Engineering, this research could have much wider application. Once expanded, the programs developed through this research could potentially be usea by all Air Force and private sector HVAC engineers who design or analyze conventional control systems.

If these designers use the CMOD programs in conjunction with the School of Civil Engineering, HVAC Control Systems design procedures, several significant benefits should result. The computer method will enable the control system engineers to analyze complete control systems in a fraction of the time required to perform the same task manually, as is currently done.

However, the real goal is to do a better job of optimizing control system designs in order to achieve energy conservation, maintainability, and all the associated benefits discussed in the background chapter of this document. Since the computer programs will enable the designer to analyze many variations of a proposed control system in a short time, the programs will significantly help the designers to achieve this goal.

Finally, the computer method has the potential to enhance the learning process which occurs in the HVAC Control Systems course. Once the students understand the manual method, they can save an enormous amount of time by using the computer method and thus leave more class hours available for "learning" as opposed to "number crunching".

Recommendations for Future Work

The prototype developed through this research is fully operational as it currently exists; however, the work is incomplete in that additional control system components need to be modeled and integrated into the computer programs. As described in Chapter V, only three control system components were modeled through this research while there are between ten to twenty "standard" components which are commonly used in conventional electronic control systems. Obviously, the remaining components need to be modeled before the programs developed through this research could be used in practice for virtually any conventional electronic proportional control system.

However, while modeling an additional fifteen or so control system components would make this method universally applicable, incorporating only five addidional components into the programs will enable the School of Civil Engineering faculty to use the programs to model two of the control systems which are analyzed in the HVAC Control Systems course. These systems are an electronic single zone system with mixed air reset and an electronic single zone system with space temperature control of the mixed air section. The additional five components which need to be modeled in order to analyze both systems include a two-input electronic proportional controller, an electronic signal sequencer, a temperature sensitive switch (commonly used as a high-limit switch), an electronic adapter that passes on the greater of a preset signal or the input signal (commonly used as a minimum position damper control) and finally, a low signal selector that passes on the lower of two signals (opposite of the high signal selector already modeled).

As discussed in the previous chapter, modeling each new component would require three subroutines and one FOR/NEXT loop. However, the computer programing logic needed to model the five new components will essentially follow the logic established in this research. Furthermore, the components can be modeled using elementry algebraic equations previously developed for the manual method. Consequently, the additional work needed can easily be accomplished by the School of Civil Engineering faculty or anyone who is familiar with the HVAC Control Systems course subject matter.

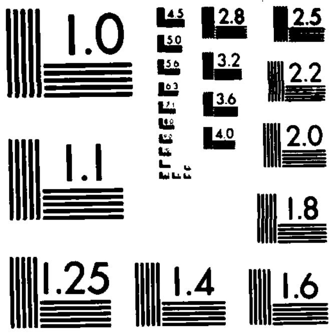
The computer programs developed through this research can also be expanded to facilitate modeling pneumatic HVAC control systems. Accomplishing this task will require both writing component-specific subroutines and slightly modifying the main programs.

Writing the pneumatic component subroutines will require the same amount of effort as writing the additional subroutines for electronic components because the AFIT manual procedures were developed for both pneumatic and electronic control systems. Consequently, the algebraic equations needed to write the subroutines for both types of components already exist.

The main programs will also have to be modified to let the user select either an electronic or a pneumatic system. In addition to prompting for the appropriate system type, these main-program modifications must incorporate provisions for passing program control to the appropriate set of subroutines (pneumatic or electronic) once the user selects either type.

In addition to incorporating additional component-specific subroutines into the CMOD programs and expanding the programs to model pneumatic systems, the research team believes it may be possible to improve one limiting feature of the CMODEDD program. As it currently exists, the user can not remove individual components from an existing data disk file using CMODEDD. Instead, the user must create a new file using CMODCRD with the desired components omitted. Clearly, it would be beneficial to be able to remove components (i.e., individual records of the data disk file)

AD-A174 617   PROTOTYPE OF A COMPUTER METHOD FOR DESIGNING AND   2/2
ANALYZING HEATING VENTIL (U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF SYST   S J BARLOW
UNCLASSIFIED   SEP 86 AFIT/GEM/DET/86S-1   F/G 13/1   NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

from an existing control system (file) rather than have to create a new system (file). Although none of the BASIC references studied by the researcher addressed how to accomplish this removal task, the researcher has not concluded that this task is impossible. The possibility of developing the appropriate computer code to remove individual records is one of the areas that any future researchers should investigate further.

Appendix A. Printout of CMOD Programs

Appendix A contains printouts of the four working computer programs and the documentation program developed under this research effort. The programs are written in the BASIC programming language. The programs appear in the following order:

CMODCRD	Create Data File Program
CMODPRD	Print Data File Program
CMODEXC	Execute Program
CMODEDD	Edit Data File Program
CMODDOC	Documentation Program

CMODCRD

```
100 PRINT: LPRINT
110 PRINT: LPRINT
120 FEW$="THE FILENAME ENTERED WAS "
130 EAC$="DO YOU WANT TO ENTER ANOTHER "
140 ITC$="IS THIS CORRECT "
150 YN$=" (YES OR NO) "
160 MNP$="ENTER NUMBER FOR APPROPRIATE COMPONENT TYPE"
170 PSTR$="PRESENTLY STORED IN THIS DATA FILE"
180 GOTO 240
190 OPEN "R",#1,RAFN$,32
200 FIELD #1,2 AS RN$,2 AS CMPC$
210 LSET RN$=MKI$(1): LSET CMPC$=MKI$(0)
220 PUT #1,1: CLOSE #1
230 GOTO 400
240 PRINT "CREATE DATA FILE PROGRAM"
250 LPRINT "CREATE DATA FILE PROGRAM"
260 PRINT "************************"
270 LPRINT "************************"
280 PRINT: LPRINT
290 PRINT: LPRINT
300 INPUT "ENTER THE DATA FILE NAME (UP TO 8 LETTERS) FOR THIS SYSTEM";RAFN$
310 LPRINT "ENTER THE DATA FILE NAME (UP TO 8 LETTERS) FOR THIS SYSTEM"
320 PRINT: LPRINT: PRINT FEW$; RAFN$: LPRINT FEW$; RAFN$: PRINT: LPRINT
330 PRINT "IS FILE NAME CORRECT ";YN$: LPRINT:IS FILE NAME CORRECT ";YN$
335 INPUT ANS$
340 PRINT: LPRINT: IF ANS$="YES" THEN 350 ELSE 300
350 PRINT "IS ";RAFN$;" A NEW OR EXISTING DATA FILENAME (NEW OR EXIST)"
360 LPRINT "IS ";RAFN$;" A NEW OR EXISTING DATA FILENAME (NEW OR EXIST)"
370 INPUT ANS$: PRINT: LPRINT
380 IF ANS$="NEW" THEN 190 ELSE 400
390 PRINT: LPRINT: PRINT: LPRINT
393 PRINT "TO LOAD COMPONENTS INTO DATA DISK FILE"
394 LPRINT "TO LOAD COMPONENTS INTO DATA DISK FILE"
395 PRINT "SELECT COMPONENTS FORM MENU BY TYPING IN APPROPRIATE NUMBER"
396 LPRINT "SELECT COMPONENTS FROM MENU BY TYPING IN APPROPRIATE NUMBER"
397 PRINT: LPRINT
400 PRINT "COMPONENT MENU": PRINT: LPRINT "COMPONENT MENU": LPRINT
410 PRINT "1 = SINGLE INPUT CONTROLLER":LPRINT"1 = SINGLE INPUT CONTROLLER"
420 PRINT "2 = TEMPERATURE SENSOR": LPRINT "2 = TEMPERATURE SENSOR"
430 PRINT "3 = HI SELECTOR": LPRINT "3 = HI SELECTOR"
440 PRINT: LPRINT
450 PRINT MNP$: INPUT X: LPRINT MNP$: PRINT: LPRINT
460 ON X GOSUB 1000,2000,3000
470 PRINT: LPRINT
480 PRINT EAC$;"COMPONENT";YN$: LPRINT EAC$;"COMPONENT";YN$: INPUT ANS$
490 IF ANS$="YES" THEN 400 ELSE 500
500 PRINT: LPRINT
501 PRINT "**CREATE** DATA FILE PROGRAM COMPLETE": PRINT
502 LPRINT "**CREATE** DATA FILE PROGRAM COMPLETE": LPRINT
503 PRINT "SELECT ANOTHER PROGRAM BY TYPING 'RUN programname'"
504 LPRINT "SELECT ANOTHER PROGRAM BY TYPING 'RUN programname'"
505 PRINT: LPRINT: PRINT: LPRINT
510 END
1000 PRINT "SINGLE INPUT CONTROLLER DATA ENTRY SUBROUTINE": PRINT
1010 LPRINT "SINGLE INPUT CONTROLLER DATA ENTRY SUBROUTINE": LPRINT
1020 OPEN "R",#1,RAFN$,32
1030 FIELD #1,2 AS RN$,2 AS CMPC$
1040 FIELD #1,2 AS N$,2 AS TYP$,5 AS SY$,4 AS OTPT$,4 AS SP$,4 AS TR$,2 AS ACT$,
5 AS INSY$,4 AS TIN$
1050 GET #1,1
1060 NR=CVI(RN$): NC=CVI(CMPC$)
1070 PRINT NC;" COMPONENTS ";PSTR$: PRINT
1080 LPRINT NC;" COMPONENTS ";PSTR$: LPRINT
1090 INPUT "ENTER ANOTHER CONTROLLER (YES OR NO)"; ANS$
1100 LPRINT "ENTER ANOTHER CONTROLLER ?"
1110 IF ANS$ = "YES" THEN 1120 ELSE 1370
1120 NC=NC+1: NR=NR+1
```

```
1121 PRINT: LPRINT
1130 PRINT "(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)"
1131 PRINT "(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)": PRINT
1140 LPRINT "(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)"
1141 LPRINT "(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)": LPRINT
1150 INPUT "CONTROLLER SYMBOL = ";SYI$: LPRINT "CONTROLLER SYMBOL = ?"
1160 INPUT "SET POINT = ";SPI: LPRINT "SET POINT = ?"
1170 INPUT "THROT RANGE = ";TRI: LPRINT "THROT RANGE = ?"
1180 INPUT "ACTION = (DA OR RA)";ACTI$: LPRINT "ACTION = (DA OR RA) ?"
1190 INPUT "SYMBOL OF INPUT COMPONENT = ";INSYI$: LPRINT "SYMBOL OF INPUT COMPON
ENT ="
1200 PRINT: LPRINT
1210 PRINT "FOR CONTROLLER ";SYI$;" - ": LPRINT "FOR CONTROLLER ";SYI$;" - "
1220 PRINT "SP = ";SPI: LPRINT "SP = ";SPI
1230 PRINT "TR = ";TRI: LPRINT "TR = ";TRI
1240 PRINT "ACT = ";ACTI$: LPRINT "ACT = ";ACTI$
1250 PRINT "INPUT COMPONENT = ";INSYI$: LPRINT "INPUT COMPONENT =";INSYI$
1260 PRINT: LPRINT:
1270 INPUT "IS THIS CORRECT (YES OR NO) ";ANS$: LPRINT "IS THIS CORRECT ?"
1280 IF ANS$ = "YES" THEN 1300
1290 PRINT "INPUT CORRECT DATA": GOTO 1130: LPRINT "INPUT CORRECT DATA"
1300 LSET N$=MKI$(NR): LSET TYP$=MKI$(1): LSET SY$=SYI$: LSET SP$=MKS$(SPI)
1310 LSET TR$=MKS$(TRI): LSET ACT$=ACTI$: LSET INSY$=INSYI$
1320 LSET OTPT$=MKS$(999): LSET TIN$=MKS$(999)
1330 PUT #1,NR
1340 LSET RN$=MKI$(NR): LSET CMPC$=MKI$(NC)
1350 PUT #1,1
1360 GOTO 1070
1370 CLOSE #1
1380 RETURN
2000 PRINT "TEMP SENSOR DATA ENTRY SUBROUTINE": PRINT
2010 LPRINT "TEMP SENSOR DATA ENTRY SUBROUTINE": LPRINT
2020 OPEN "R",#1,RAFN$,32
2030 FIELD #1,2 AS RN$,2 AS CMPC$
2040 FIELD #1,2 AS N$,2 AS TYP$, 5 AS SY$,4 AS OTPT$, 4 AS STIN$
2050 GET #1,1
2060 NR=CVI(RN$): NC=CVI(CMPC$)
2070 PRINT NC;" COMPONENTS ";PSTR$: PRINT
2080 LPRINT NC;" COMPONENTS ";PSTR$: LPRINT
2090 PRINT EAC$;"SENSOR";YN$: LPRINT EAC$;"SENSOR";YN$: INPUT ANS$
2100 IF ANS$="YES" THEN 2110 ELSE 2270
2110 NC=NC+1: NR=NR+1: PRINT: LPRINT
2120 PRINT "(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)"
2121 PRINT "(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)": PRINT
2122 LPRINT "(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)"
2123 LPRINT "(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)": LPRINT
2130 INPUT "SENSOR SYMBOL = ";SYI$: LPRINT "SENSOR SYMBOL = "
2140 INPUT "TEMP IN = ";STINI: LPRINT "TEMP IN = "
2150 STOUTI=STINI: PRINT: LPRINT
2160 PRINT "FOR SENSOR ";SYI$;" TEMP IN = ";STINI
2170 LPRINT "FOR SENSOR ";SYI$;" TEMP IN = ";STINI
2180 INPUT "IS THIS CORRECT (YES OR NO)";ANS$: LPRINT "IS THIS CORRECT"
2190 PRINT: LPRINT: IF ANS$="YES" THEN 2210
2200 PRINT "INPUT CORRECT DATA": LPRINT "INPUT CORRECT DATA": GOTO 2120
2210 LSET N$=MKI$(NR): LSET TYP$=MKI$(2): LSET SY$=SYI$:LSET OTPT$=MKS$(STOUTI)
2220 LSET STIN$=MKS$(STINI)
2230 PUT #1,NR
2240 LSET RN$=MKI$(NR): LSET CMPC$=MKI$(NC)
2250 PUT #1,1
2260 PRINT: LPRINT: GOTO 2070
2270 CLOSE #1
2280 RETURN
3000 PRINT "HI SELECTOR DATA ENTRY SUBROUTINE": PRINT
3010 LPRINT "HI SELECTOR DATA ENTRY SUBROUTINE": LPRINT
3020 OPEN "R",#1,RAFN$,32
3030 FIELD #1,2 AS RN$,2 AS CMPC$
3040 FIELD #1,2 AS N$,2 AS TYP$,5 AS SY$,4 AS OTPT$,5 AS INY1$,4 AS HIN1$,5 AS I
NY2$,4 AS HIN2$
3050 GET #1,1
3060 NR=CVI(RN$): NC=CVI(CMPC$)
3070 PRINT NC;" COMPONENTS ";PSTR$: PRINT
```

```
3080 LPRINT NC;" COMPONENTS ";PSTR$: LPRINT
3090 INPUT "ENTER ANOTHER HI SELECTOR (YES OR NO)";ANS$
3100 LPRINT "ENTER ANOTHER HI SELECTOR"
3110 IF ANS$="YES" THEN 3120 ELSE 3410
3120 NC=NC+1: NR=NR+1
3130 PRINT: LPRINT
3140 PRINT "(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)"
3150 LPRINT "(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)"
3160 PRINT "(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)"
3170 LPRINT "(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)"
3180 PRINT: LPRINT
3190 INPUT "HI SELECTOR SYMBOL = ";SYI$: LPRINT "HI SELECTOR SYMBOL ="
3200 INPUT "SYMBOL OF 1ST INPUT COMPONENT = ";INY1I$:
3210 LPRINT "SYMBOL OF 1ST INPUT COMPONENT ="
3220 INPUT "SYMBOL OF 2ND INPUT COMPONENT = ";INY2I$
3230 LPRINT "SYMBOL OF 2ND INPUT COMPONENT ="
3240 PRINT: LPRINT
3250 PRINT "FOR HI SELECTOR ";SYI$;" - ": LPRINT "FOR HI SELECTOR ";SYI$;" - "
3260 PRINT "SYMBOL OF 1ST INPUT COMPONENT = ";INY1I$
3270 LPRINT "SYMBOL OF 1ST INPUT COMPONENT = ";INY1I$
3280 PRINT "SYMBOL OF 2ND INPUT COMPONENT = ";INY2I$
3290 LPRINT "SYMBOL OF 2ND INPUT COMPONENT = ";INY2I$
3300 PRINT: LPRINT
3310 INPUT "IS THIS CORRECT (YES OR NO) ";ANS$: LPRINT "IS THIS CORRECT"
3320 IF ANS$="YES" THEN 3340
3330 PRINT "INPUT CORRECT DATA": GOTO 3190: LPRINT "INPUT CORRECT DATA"
3340 LSET N$=MKI$(NR): LSET TYP$=MKI$(3): LSET SY$=SYI$: LSET INY1$=INY1I$
3350 LSET INY2$=INY2I$
3360 LSET OTPT$=MKS$(999): LSET HIN1$=MKS$(999): LSET HIN2$=MKS$(999)
3370 PUT #1,NR
3380 LSET RN$=MKI$(NR): LSET CMPC$=MKI$(NC)
3390 PUT #1,1
3400 GOTO 3070
3410 CLOSE #1
3420 RETURN
```

CMODPRD

```
100 PRINT: LPRINT
110 PRINT "PRINT CONTENTS OF DATA FILE PROGRAM"
120 PRINT "***********************************"
130 LPRINT "PRINT CONTENTS OF DATA FILE PROGRAM"
140 LPRINT "***********************************"
150 PRINT: LPRINT
160 INPUT "ENTER DATA FILE NAME (UP TO 8 LETTERS) FOR THIS SYSTEM";RAFN$
170 LPRINT "ENTER DATA FILE NAME (UP TO 8 LETTERS) FOR THIS SYSTEM"
180 PRINT "THE FILENAME ENTERED WAS ";RAFN$: INPUT "IS THIS CORRECT";ANS$
190 LPRINT "THE FILENAME ENTERED WAS ";RAFN$
200 LPRINT "IS THIS CORRECT (YES OR NO)"
210 PRINT: LPRINT
220 IF ANS$="YES" THEN 230 ELSE 160
230 PRINT: LPRINT
240 OPEN "R",#1,RAFN$,32
250 FIELD #1,2 AS RN$,2 AS CMPC$
260 FIELD #1,2 AS N$,2 AS TYP$,5 AS SY$,4 AS OTPT$,4 AS SP$,4 AS TR$,2 AS ACT$,5
 AS INSY$,4 AS TIN$
270 FIELD #1,2 AS N$,2 AS TYP$,5 AS SY$,4 AS OTPT$,4 AS STIN$
280 FIELD #1,2 AS N$,2 AS TYP$,5 AS SY$,4 AS OTPT$,5 AS INY1$,4 AS HIN1$,5 AS IN
Y2$,4 AS HIN2$
290 REM
300 PRINT: PRINT: LPRINT: LPRINT
310 PRINT "DATA FOR CONTROLLERS": LPRINT "DATA FOR CONTROLLERS"
320 PRINT: LPRINT
330 PRINT "SYMBOL  ACTION  THROT RANGE  SET POINT  INPUT COMPONENT";
340 PRINT "  TEMP IN  VOLT OUT"
350 LPRINT "SYMBOL  ACTION  THROT RANGE  SET POINT  INPUT COMPONENT";
```

```
360 LPRINT "  TEMP IN  VOLT OUT"
370 PRINT: LPRINT
380 GET #1,1
390 FOR C=2 TO CVI(RN$)
400 GET #1,C
410 IF CVI(TYP$)=1 THEN 420 ELSE 440
420 PRINT SY$;"   ";ACT$;"      ";CVS(TR$);"             ";CVS(SP$);"        ";I
NSY$;"         ";CVS(TIN$);"     ";CVS(OTPT$)
430 LPRINT SY$;"   ";ACT$;"      ";CVS(TR$);"             ";CVS(SP$);"        ";
INSY$;"         ";CVS(TIN$);"     ";CVS(OTPT$)
440 NEXT C
450 PRINT: PRINT: LPRINT: LPRINT
460 PRINT "DATA FOR SENSORS": LPRINT "DATA FOR SENSORS"
470 PRINT: LPRINT
480 PRINT "SYMBOL  TEMP IN  OUTPUT": PRINT
490 LPRINT "SYMBOL  TEMP IN  OUTPUT": LPRINT
500 FOR C=2 TO CVI(RN$)
510 GET #1,C
520 IF CVI(TYP$)=2 THEN 530 ELSE 550
530 PRINT SY$;"   ";CVS(STIN$);"      ";CVS(OTPT$)
540 LPRINT SY$;"   ";CVS(STIN$);"      ";CVS(OTPT$)
550 NEXT C
560 PRINT: PRINT: LPRINT: LPRINT
570 PRINT "DATA FOR HI SELECTORS": LPRINT "DATA FOR HI SELECTORS"
580 PRINT: LPRINT
590 PRINT "SYMBOL  1ST INPUT SYMBOL   SYMBOL 1 IN   2ND INPUT SYMBOL   SYMBOL 2
IN  OUTPUT"
600 LPRINT "SYMBOL  1ST INPUT SYMBOL   SYMBOL 1 IN   2ND INPUT SYMBOL   SYMBOL 2
 IN  OUTPUT"
610 PRINT: LPRINT
620 FOR C=2 TO CVI(RN$)
630 GET #1,C
640 IF CVI(TYP$)=3 THEN 650 ELSE 670
650 PRINT SY$;"      ";INY1$;"             ";CVS(HIN1$);"          ";INY2$;"
        ";CVS(HIN2$);"     ";CVS(OTPT$)
660 LPRINT SY$;"      ";INY1$;"             ";CVS(HIN1$);"          ";INY2$;"
        ";CVS(HIN2$);"     ";CVS(OTPT$)
670 NEXT C
680 CLOSE #1
690 PRINT: LPRINT
700 PRINT "**PRINT** PROGRAM RUN COMPLETE"
710 LPRINT "**PRINT** PROGRAM RUN COMPLETE"
720 PRINT: LPRINT
730 PRINT "SELECT ANOTHER PROGRAM BY TYPING RUN ''programname''(DOUBLE QUOTES)"
740 LPRINT "SELECT ANOTHER PROGRAM BY TYPING RUN ''programname''(DOUBLE QUOTES)"
750 PRINT: LPRINT
760 PRINT: LPRINT
770 END
```

CMODEXC

```
100 PRINT: PRINT: LPRINT: LPRINT
110 PRINT "EXECUTION MODE PROGRAM"
120 PRINT "**********************"
130 LPRINT "EXECUTION MODE PROGRAM"
140 LPRINT "**********************"
150 PRINT: LPRINT
160 INPUT "ENTER THE DATA FILE NAME (UP TO 8 LETTERS)"; RAFN$
170 LPRINT "ENTER THE DATA FILE NAME (UP TO 8 LETTERS)"
180 PRINT: LPRINT
190 PRINT "THE FILE NAME ENTERED WAS ";RAFN$
```

```
200 LPRINT "THE FILE NAME ENTERED WAS ";RAFN$
210 INPUT "IS FILE NAME CORRECT (YES OR NO) ";ANS$
220 LPRINT "IS FILE NAME CORRECT (YES OR NO) ?"
230 PRINT: LPRINT
240 IF ANS$="YES" THEN 250 ELSE 160
250 OPEN "R",#1,RAFN$,32
260 FIELD #1,2 AS RN$,2 AS CMPC$
270 FIELD #1,2 AS N$,2 AS TYP$,5 AS SY$,4 AS OTPT$,4 AS SP$,4 AS TR$,2 AS ACT$,5
 AS INSY$,4 AS TIN$
280 FIELD #1,2 AS N$,2 AS TYP$,5 AS SY$,4 AS OTPT$,4 AS STIN$
290 FIELD #1,2 AS N$,2 AS TYP$,5 AS SY$,4 AS OTPT$,5 AS INY1$,4 AS HIN1$,5 AS IN
Y2$,4 AS HIN2$
300 GET #1,1
310 K=CVI(RN$): L=CVI(CMPC$)
320 ITER=0
330 FOR I=2 TO K
340 GET #1,I
350 TYN=CVI(TYP$)
360 ON TYN GOSUB 1000,2000,3000
370 NEXT I
380 ITER=ITER + 1
390 PRINT ITER;" ITERATION(S) COMPLETE "
400 LPRINT ITER;" ITERATION(S) COMPLETE "
410 IF ITER < L THEN 330 ELSE 420
420 PRINT "FOR THIS ";L;" COMPONENT SYSTEM "
430 LPRINT "FOR THIS ";L;" COMPONENT SYSTEM "
440 PRINT: LPRINT
450 PRINT: LPRINT
460 CLOSE #1
470 PRINT: LPRINT
480 PRINT "**EXECUTION** PROGRAM RUN COMPLETE"
490 LPRINT "**EXECUTION** PROGRAM RUN COMPLETE"
500 PRINT: LPRINT
510 PRINT "RUN CMODPRD TO SEE RESULTS OF EXECUTION"
520 LPRINT "RUN CMODPRD TO SEE RESULTS OF EXECUTION"
530 PRINT: LPRINT: PRINT: LPRINT
540 END
1000 REM ** SINGLE INPUT CONTROLLER EXECUTION SUBROUTINE **
1010 SPT=CVS(SP$): TRG=CVS(TR$): AC$=ACT$: SYM$=INSY$
1020 FOR J=2 TO K
1030 IF J<>I THEN 1040 ELSE 1080
1040 GET #1,J
1050 IF SY$=SYM$ THEN 1060 ELSE 1080
1060 T=CVS(OTPT$)
1070 GOTO 1090
1080 NEXT J
1090 IF AC$="DA" THEN 1100 ELSE 1120
1100 V=7.5 + 3*((T-SPT)/TRG)
1110 GOTO 1130
1120 V=7.5 - 3*((T-SPT)/TRG)
1130 REM
1140 GET #1,I
1150 LSET TIN$=MKS$(T): LSET OTPT$=MKS$(V)
1160 PUT #1,I
1170 RETURN
2000 REM ** TEMPERATURE SENSOR EXECUTION SUBROUTINE **
2010 REM          NO CALCULATIONS NEEDED
2020 RETURN
3000 REM ** HI SELECTOR EXECUTION SUBROUTINE **
3010 INSYM1$=INY1$: INSYM2$=INY2$
3020 FOR J=2 TO K
3030 GET #1,J
3040 IF SY$=INSYM1$ THEN 3050 ELSE 3070
3050 HINP1=CVS(OTPT$)
3060 GOTO 3090
3070 IF SY$=INSYM2$ THEN 3080 ELSE 3090
3080 HINP2=CVS(OTPT$)
```

```
3090 NEXT J
3100 GET #1,I
3110 LSET HIN1$=MKS$(HINP1): LSET HIN2$=MKS$(HINP2)
3120 IF HINP1 > HINP2 THEN 3130 ELSE 3140
3130 LSET OTPT$=MKS$(HINP1): GOTO 3150
3140 LSET OTPT$=MKS$(HINP2)
3150 PUT #1,I
3160 RETURN
```

CMODEDD

```
100 PRINT: PRINT: LPRINT: LPRINT
110 PRINT "DATA EDIT PROGRAM": LPRINT "DATA EDIT PROGRAM"
120 PRINT "*****************": LPRINT "*****************"
130 PRINT: LPRINT
140 PRINT "SELECT OPTION BY TYPING IN NUMBER INDICATED ON MENU"
150 LPRINT "SELECT OPTION BY TYPING IN THE NUMBER INDICATED ON MENU"
160 PRINT "                                        MENU"
170 LPRINT "                                        MENU"
180 PRINT: LPRINT
190 PRINT "                  OPTION                                    NUMBER"
200 LPRINT "                  OPTION                                    NUMBER"
210 PRINT: LPRINT
220 PRINT "EDIT CONTENTS OF EXISTING DATA FILE................. 1"
230 LPRINT "EDIT CONTENTS OF EXISTING DATA FILE................. 1"
240 PRINT "ADD ADDITIONAL COMPONENTS TO EXISTING DATA FILE..... 2"
250 LPRINT "ADD ADDITIONAL COMPONENTS TO EXISTING DATA FILE..... 2"
260 PRINT "REMOVE COMPONENTS FROM EXISTING DATA FILE........... 3"
270 LPRINT "REMOVE COMPONENTS FROM EXISTING DATA FILE........... 3"
280 PRINT: LPRINT
290 INPUT NUM
300 IF NUM = 2 THEN 630 ELSE 305
305 IF NUM = 3 THEN 660 ELSE 310
310 INPUT "ENTER THE DATA FILE NAME (UP TO 8 LETTERS)"; RAFN$
320 LPRINT "ENTER THE DATA FILE NAME (UP TO 8 LETTERS)"
330 PRINT "THE FILE NAME ENTERED WAS ";RAFN$
340 LPRINT "THE FILE NAME ENTERED WAS ";RAFN$
350 INPUT "IS FILE NAME CORRECT (YES OR NO) ";ANS$
360 LPRINT "IS FILE NAME CORRECT (YES OR NO) ?"
370 PRINT: LPRINT
380 IF ANS$="YES" THEN 390 ELSE 310
390 '
400 OPEN "R",#1,RAFN$,32
410 FIELD #1,2 AS RN$,2 AS CMPC$
420 FIELD #1,2 AS N$,2 AS TYP$,5 AS SY$,4 AS OTPT$,4 AS SP$,4 AS TR$,2 AS ACT$,5
 AS INSY$,4 AS TIN$
430 FIELD #1,2 AS N$,2 AS TYP$,5 AS SY$,4 AS OTPT$,4 AS STIN$
440 FIELD #1,2 AS N$,2 AS TYP$,5 AS SY$,4 AS OTPT$,5 AS INY1$,4 AS HIN1$,5 AS IN
Y2$,4 AS HIN2$
450 '
460 PRINT "(REMEMBER TO USE *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)"
470 LPRINT "(REMEMBER TO USE *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)"
480 INPUT "ENTER SYMBOL OF COMPONENT YOU WISH TO EDIT";SYI$
490 GET #1,1
500 K=CVI(RN$)
510 FOR I=2 TO K
520 GET #1,I
530 IF SY$=SYI$ THEN 550 ELSE 540
540 NEXT I
550 TYN=CVI(TYP$)
560 ON TYN GOSUB 1000,2000,3000
570 PRINT: INPUT "DO YOU WANT TO EDIT ANOTHER COMPONENT (YES OR NO)";ANS$
```

```
580 LPRINT: LPRINT "DO YOU WANT TO EDIT ANBOTHER COMPONENT  (YES OR NO) ?"
590 PRINT: LPRINT
600 IF ANS$="YES" GOTO 460
610 CLOSE #1
620 GOTO 720
630 PRINT "TO ADD NEW COMPONENTS TO DATA FILE, RUN 'CMODCRD' "
640 LPRINT "TO ADD NEW COMPONENTS TO DATA FILE, RUN 'CMODCRD'"
650 PRINT: LPRINT: GOTO 720
660 PRINT "TO REMOVE COMPONENTS FROM EXISTING DATA FILE"
670 PRINT "THE USER MUST CREATE A NEW DATA FILE WITH THE UNDESIRED"
680 PRINT "COMPONENTS OMITTED"
690 LPRINT "TO REMOVE COMPONENTS FROM EXISTING DATA FILE"
700 LPRINT "THE USER MUST CREATE A NEW DATA FILE WITH THE UNDESIRED"
710 LPRINT "COMPONENTS OMITTED"
720 PRINT: LPRINT
730 PRINT "**EDIT** PROGRAM RUN COMPLETE"
740 LPRINT "**EDIT** PROGRAM RUN COMPLETE"
750 PRINT: LPRINT
760 PRINT "RUN CMODEXC TO UPDATE DATA FILE TO REFLECT NEW OUTPUTS"
770 LPRINT "RUN CMODEXC TO UPDATE DATA FILE TO REFLECT NEW OUTPUTS"
780 PRINT: LPRINT: PRINT: LPRINT
790 END
1000 PRINT: PRINT: LPRINT: LPRINT
1001 PRINT "SINGLE INPUT CONTROLLER **EDIT** SUBROUTINE": PRINT
1010 LPRINT "SINGLE INPUT CONTROLLER **EDIT** SUBROUTINE": LPRINT
1020 OUTS=CVS(OTPT$): SPI=CVS(SP$): TRI=CVS(TR$): ACTI$=ACT$
1030 INSYI$=INSY$: TINI=CVS(TIN$)
1040 PRINT "THE DATA PRESENTLY STORED FOR ";SYI$;" IS ": PRINT
1050 LPRINT "THE DATA PRESENTLY STORED FOR ";SYI$;" IS ": LPRINT
1060 PRINT "SP = ";SPI: LPRINT "SP = ";SPI
1070 PRINT "TR = ";TRI: LPRINT "TR = ";TRI
1080 PRINT "ACT = ";ACTI$: LPRINT "ACT = ";ACTI$
1090 PRINT "INPUT COMPONENT = ";INSYI$: LPRINT "INPUT COMPONENT = ";INSYI$
1100 PRINT "TEMP IN = ";TINI: LPRINT "TEMP IN = ";TINI
1110 PRINT "OUTPUT = ";OUTS: LPRINT "OUTPUT = ";OUTS
1120 PRINT: LPRINT
1130 PRINT: LPRINT
1140 PRINT "SELECT ITEM YOU WISH TO CHANGE FROM MENU"
1150 LPRINT "SELECT ITEM YOU WISH TO CHANGE FROM MENU"
1160 PRINT
1170 LPRINT
1180 PRINT "                       MENU"
1190 LPRINT"                        MENU"
1200 PRINT"TO CHANGE :                     TYPE IN :"
1210 LPRINT"TO CHANGE :                     TYPE IN :"
1220 PRINT "  SET POINT                       1"
1230 LPRINT "  SET POINT                       1"
1240 PRINT "  THROT RANGE                     2"
1250 LPRINT "  THROT RANGE                     2"
1260 PRINT "  ACTION                          3"
1270 LPRINT "  ACTION                          3"
1280 PRINT "  INPUT SYMBOL                    4"
1290 LPRINT "  INPUT SYMBOL                    4"
1300 PRINT
1310 LPRINT
1320 PRINT "TO CHANGE THE VALUE OF 'TEMP IN', YOU MUST 'EDIT' ";INSYI$
1330 LPRINT "TO CHANGE THE VALUE OF 'TEMP IN', YOU MUST 'EDIT' ";INSYI$
1340 PRINT
1350 LPRINT
1360 INPUT "TYPE IN APPROPRIATE NUMBER";T
1370 LPRINT "TYPE IN APPROPRIATE NUMBER"
1380 ON T GOSUB 1400,1440,1480,1520
1390 GOTO 1560
1400 INPUT "NEW SET POINT = ";SPI: LPRINT "NEW SET POINT ="
1410 LSET SP$=MKS$(SPI)
1420 PUT #1,I
1430 RETURN
1440 INPUT "NEW THROT RANGE = ";TRI: LPRINT "NEW THROT RANGE = "
```

```
1450 LSET TR$=MKS$(TRI)
1460 PUT #1,I
1470 RETURN
1480 INPUT "NEW ACTION = ";ACTI$: LPRINT "NEW ACTION ="
1490 LSET ACT$=ACTI$
1500 PUT #1,I
1510 RETURN
1520 INPUT "NEW INPUT SYMBOL = ";INSYI$: LPRINT "NEW INPUT SYMBOL ="
1530 LSET INSY$=INSYI$
1540 PUT #1,I
1550 RETURN
1560 RETURN
2000 PRINT: PRINT: LPRINT: LPRINT
2001 PRINT "TEMPERATURE SENSOR **EDIT** SUBROUTINE": PRINT
2010 LPRINT "TEMPERATURE SENSOR **EDIT** SUBROUTINE": LPRINT
2020 OUTS=CVS(OTPT$): STINI=CVS(STIN$)
2030 PRINT "THE DATA PRESENTLY STORED FOR ";SYI$;" IS ": PRINT
2040 LPRINT "THE DATA PRESENTLY STORED FOR ";SYI$;" IS ": LPRINT
2050 PRINT "TEMP IN = ";STINI
2060 LPRINT "TEMP IN = ";STINI
2070 PRINT "OUTPUT = ";OUTS
2080 LPRINT "OUTPUT = ";OUTS
2090 PRINT: LPRINT
2100 PRINT "THE ONLY ITEM YOU CAN CHANGE IS 'TEMP IN'"
2110 LPRINT "THE ONLY ITEM YOU CAN CHANGE IS 'TEMP IN'"
2120 PRINT: LPRINT
2130 INPUT "NEW VALUE FOR TEMP IN = ";STINI
2140 LPRINT "NEW VALUE FOR TEMP IN ="
2150 OUTS = STINI
2160 LSET STIN$=MKS$(STINI): LSET OTPT$=MKS$(OUTS)
2170 PUT #1,I
2180 RETURN
3000 PRINT: PRINT: LPRINT: LPRINT
3010 PRINT "HI SELECTOR **EDIT** SUBROUTINE": PRINT
3020 LPRINT "HI SELECTOR **EDIT** SUBROUTINE": LPRINT
3030 OUTS=CVS(OTPT$): INY1I$=INY1$: HIN1I=CVS(HIN1$)
3040 INY2I$=INY2$: HIN2I=CVS(HIN2$)
3050 PRINT "THE DATA PRESENTLY STORED FOR ";SYI$;" IS ": PRINT
3060 LPRINT "THE DATA PRESENTLY STORED FOR ";SYI$;" IS ": LPRINT
3070 PRINT "1ST INPUT COMPONENT = ";INY1I$
3080 LPRINT "1ST INPUT COMPONENT = ";INY1I#
3090 PRINT "INPUT SIGNAL FROM ";INY1I$;" = ";HIN1I
3100 LPRINT "INPUT SIGNAL FROM ";INY1I$;" = ";HIN1I
3110 PRINT "2ND INPUT COMPONENT = ";INY2I$
3120 LPRINT "2ND INPUT COMPONENT = ";INY2I$
3130 PRINT "INPUT SIGNAL FROM ";INY2I$;" = ";HIN2I
3140 LPRINT "INPUT SIGNAL FROM ";INY2I$;" = ";HIN2I
3150 PRINT "OUTPUT SIGNAL = ";OUTS
3160 LPRINT "OUTPUT SIGNAL = ";OUTS
3170 PRINT: PRINT
3180 LPRINT: LPRINT
3190 PRINT "SELECT ITEM YOU WISH TO CHANGE FROM MENU"
3200 LPRINT "SELECT ITEM YOU WISH TO CHANGE FROM MENU"
3210 PRINT: LPRINT
3220 PRINT "                          MENU"
3230 LPRINT "                          MENU"
3240 PRINT "   TO CHANGE :                        TYPE IN :"
3250 LPRINT "   TO CHANGE :                       TYPE IN :"
3260 PRINT "1ST INPUT COMPONENT SYMBOL        1"
3270 LPRINT "1ST INPUT COMPONENT SYMBOL        1"
3280 PRINT "2ND INPUT COMPONENT SYMBOL        2"
3290 LPRINT "2ND INPUT COMPONENT SYMBOL        2"
3300 PRINT: LPRINT
3310 PRINT "TO CHANGE THE VALUE OF ";INY1I$;" YOU MUST 'EDIT' ";INY1I$
3320 LPRINT "TO CHANGE THE VALUE OF ";INY1I$;" YOU MUST 'EDIT' ";INY1I$
3330 PRINT: LPRINT
3340 PRINT "TO CHANGE THE VALUE OF ";INY2I$;" YOU MUST 'EDIT' ";INY2I$
3350 LPRINT "TO CHANGE THE VALUE OF ";INY2I$;" YOU MUST 'EDIT' ";INY2I$
```

```
3360 PRINT: LPRINT
3370 PRINT: LPRINT
3380 INPUT "TYPE IN APPROPRIATE NUMBER";T
3390 LPRINT "TYPE IN APPROPRIATE NUMBER"
3400 ON T GOSUB 3420,3470
3410 GOTO 3520
3420 INPUT "NEW SYMBOL OF 1ST INPUT COMPONENT = ";INY1I$
3430 LPRINT "NEW SYMBOL OF 1ST INPUT COMPONENT = "
3440 LSET INY1$=INY1I$
3450 PUT #1,I
3460 RETURN
3470 INPUT "NEW SYMBOL OF 2ND INPUT COMPONENT = ";INY2I$
3480 LPRINT "NEW SYMBOL OF 2ND INPUT COMPONENT = ";INY2I$
3490 LSET INY2$=INY2I$
3500 PUT #1,I
3510 RETURN
3520 RETURN
```

CMODDOC

```
100 PRINT"          WELCOME TO CONTROL MODEL (CMOD)"
110 PRINT
120 PRINT"INTRODUCTION"
130 PRINT"************"
140 PRINT"CMOD IS A PROTOTYPE OF A SERIES OF 5 PROGRAMS, A DOCUMENTATION"
150 PRINT"PROGRAM AND 4 WORKING PROGRAMS, WRITTEN IN BASIC."  THE PURPOSE"
160 PRINT"OF CMOD IS TO MODEL EXISTING CONVENTIONAL HVAC PROPORTIONAL CONTROL"
170 PRINT"SYSTEMS THAT ARE CURRENTLY USED IN NEARLY ALL AIR FORCE FACILITIES"
180 PRINT"AND AN OVERWHELMING NUMBER OF PRIVATE SECTOR FACILITIES AS WELL"
190 PRINT
200 PRINT"THROUGH THE CMOD PROGRAMS, THE USER 'LOADS' A CONTROL SYSTEM INTO"
210 PRINT"THE COMPUTER AND SPECIFIES THE SYSTEM INPUTS.  THE COMPUTER THEN"
220 PRINT"CALCULATES OUTPUTS FOR EACH COMPONENT OF THE CONTROL SYSTEM.  THUS"
230 PRINT"THE USER CAN SEE IF A CONTROL SYSTEM IS OPERATING AS INTENDED OR"
240 PRINT"HE CAN PREDICT HOW ALL COMPONENTS OF A CONTROL SYSTEM SHOULD"
250 PRINT"RESPOND UNDER THE FULL RANGE OF OPERATING CONDITIONS IN A FRAC-"
260 PRINT"TION OF THE TIME IT WOULD TAKE TO PERFORM THIS PROCESS MANUALLY."
270 PRINT: PRINT
280 INPUT"   TYPE IN ANY NUMBER, THEN HIT RETURN, TO CONTINUE";N
290 ON N GOTO 300
300 PRINT
310 PRINT"MOTIVATION"
320 PRINT"**********"
330 PRINT"THE MOTIVATION FOR THE CMOD SERIES WAS TO DEVELOP A COMPUTERIZED"
340 PRINT"METHODOLOGY TO ENHANCE THE GENERIC, HVAC CONTROL SYSTEM MANUAL"
350 PRINT"DESIGN/ANALYSIS METHODOLOGY DEVELOPED AT THE AIR FORCE INSTITUTE"
360 PRINT"OF TECHNOLOGY, SCHOOL OF CIVIL ENGINEERING, WRIGHT PATTERSON AFB,"
370 PRINT"OHIO."
380 PRINT
390 PRINT"OVERVIEW"
400 PRINT"********"
410 PRINT"THE FIVE PROGRAMS OF CMOD AND THEIR FUNCTIONS ARE AS FOLLOWS:"
420 PRINT
430 PRINT"CMODDOC  (for Control MODel DOCumentation)"
440 PRINT"******************************************"
450 PRINT"THIS IS THE INSTRUCTIONAL/DOCUMENTATION PROGRAM WHICH YOU'RE"
460 PRINT"PRESENTLY RUNNING.  IT EXPLAINS THE OVERALL SYSTEM ARCHITECTURE "
470 PRINT"AND MORE SPECIFICALLY, THE PURPOSE AND FUNCTION OF EACH OF THE FOUR"
480 PRINT"WORKING PROGRAMS:  CMODCRD, CMODPRD, CMODEXC, AND CMODEDD."
490 PRINT: PRINT
500 INPUT "   TYPE IN ANY NUMBER, THEN HIT RETURN, TO CONTINUE";N
510 ON N GOTO 520
```

```
520 PRINT
530 PRINT"CMODCRD  (for Control MODel CReate Data)"
540 PRINT"*****************************************"
550 PRINT"THIS PROGRAM IS USED TO CREATE A PERMANENT DATA FILE ON A FLOPPY "
560 PRINT"DISK WHICH CONTAINS ALL INFORMATION ABOUT THE SPECIFIC CONTROL "
570 PRINT"SYSTEM MODELLED.  IT IS ALSO USED TO ADD ADDITIONAL COMPONENTS TO"
580 PRINT"AN EXISTING DATA FILE."
590 PRINT
600 PRINT"CMODEXC  (for Control MODel EXeCute)"
610 PRINT"*************************************"
620 PRINT"THIS PROGRAM USES THE PROGRAMMING INSTRUCTIONS SPECIFIED IN CMODCRD,"
630 PRINT"COMPUTES AS NECESSARY TO PROVIDE THE INPUT AND OUTPUT SIGNAL OF"
640 PRINT"EACH CONTROL COMPONENT AND UPDATES THE DATA DISK FILE ACCORDINGLY."
650 PRINT
660 PRINT"CMODPRD  (for Control MODel PRint Data)"
670 PRINT"****************************************"
680 PRINT"THIS PROGRAM PRINTS THE CONTENTS OF THE DATA DISK FILE ON THE SCREEN"
690 PRINT"AND ON THE PRINTER."
700 PRINT: PRINT
710 INPUT"   TYPE IN ANY NUMBER, THEN HIT RETURN, TO CONTINUE";N
720 ON N GOTO 730
730 PRINT
740 PRINT"CMODEDD  (for Control MODel EDit Data)"
750 PRINT"***************************************"
760 PRINT"THIS PROGRAM EDITS THE CONTENTS OF AN EXISTING DATA DISK FILE.  IT"
770 PRINT"IS GENERALLY USED IN CONJUNCTION WITH CMODEXC TO OPTIMIZE A CONTROLS"
780 PRINT"SYSTEM DESIGN BY ALLOWING THE USER TO CHANGE SPECIFIC SYSTEM "
790 PRINT"PARAMETERS (SUCH AS A CONTROLLER ACTION) AND THEN RE-EXECUTING"
800 PRINT"AND OBSERVING THE NEW INPUTS/OUTPUTS."
810 PRINT
820 PRINT"GETTING STARTED"
830 PRINT"***************"
840 PRINT"EACH OF THE CMOD PROGRAMS IS INDEPENDENT AND CAN BE RUN ALONE OR IN"
850 PRINT"CONJUNCTION WITH ANY OF THE OTHER PROGRAMS.  ALL OF THE CMOD"
860 PRINT"PROGRAMS ARE INTERACTIVE.  ONCE THE USER SELECTS THE PROGRAM AND "
870 PRINT"TYPES RUN ''programname'', THE COMPUTER VERIFIES WHICH PROGRAM IT IS"
880 PRINT"RUNNING AND THEN BEGINS WITH A QUESTION/ANSWER SESSION.  THE USER"
890 PRINT"SIMPLY ANSWERS THE COMPUTERS PROMPTS UNTIL HE OBTAINS THE DESIRED"
900 PRINT"RESULTS."
910 PRINT: PRINT
920 INPUT"   TYPE IN ANY NUMBER, THEN HIT RETURN, TO CONTINUE";N
930 ON N GOTO 940
940 PRINT
950 PRINT"TYPICAL SESSION"
960 PRINT"***************"
970 PRINT"  1. CREATE A DATA FILE"
980 PRINT"TYPICALLY, THE FIRST THING A USER MUST DO IS CREATE A DATA DISK"
990 PRINT"FILE OF A HVAC CONTROL SYSTEM USING CMODCRD. "
1000 '
1010 PRINT
1020 PRINT"HAVING SELECTED CMODCRD BY TYPING ''RUN CMODCRD'', THE USER FOLLOWS"
1030 PRINT"THE COMPUTER PROMPTS AND TYPES IN SHORT, ONE OR TWO-WORD ANSWERS"
1040 PRINT"OR TYPES IN A NUMBER SELECTED FROM A SCREEN-DISPLAYED MENU.  THUS"
1050 PRINT"THE USER WOULD LAY OUT A SCHEMATIC OF THE HVAC CONTROL SYSTEM TO"
1060 PRINT"BE MODELLED, PROVIDE A UNIQUE NAME FOR THIS SPECIFIC SYSTEM (FILE),"
1070 PRINT"AND ENTER EACH COMPONENT AND ALL ITS ASSOCIATED PROGRAMMING "
1080 PRINT"INSTRUCTIONS ONE-AT-A-TIME, COMPONENT-BY-COMPONENT."
1090 PRINT
1100 PRINT"THE USER IS ENCOURAGED TO USE COMPONENT SYMBOLS DIRECTLY AS THEY"
1110 PRINT"APPEAR ON THE SCHEMATIC AND THEN ADD ASTERISKS TO THE END SUCH"
1120 PRINT"THAT ALL COMPONENT SYMBOLS ARE FIVE (5) CHARACTERS IN LENGTH."
1130 PRINT
1140 PRINT"FOR EXAMPLE, THE COMPUTER WILL ASK:"
1150 PRINT
1160 PRINT"     CONTROLLER SYMBOL = ?"
1170 PRINT: PRINT
1180 INPUT "   TYPE IN ANY NUMBER, THEN HIT RETURN, TO CONTINUE";N
1190 ON N GOTO 1200
```

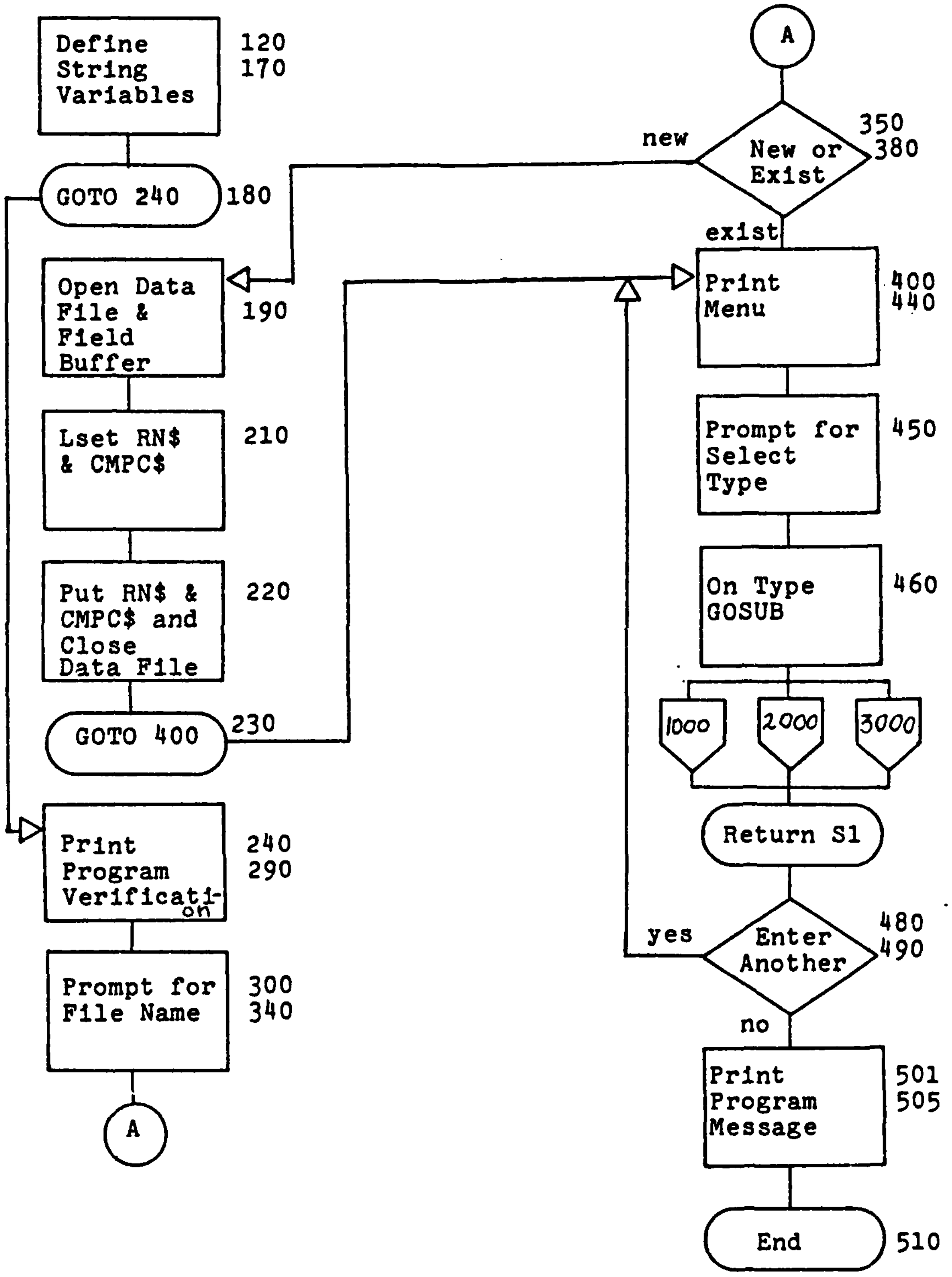
```
1200 PRINT
1210 PRINT"AND THE USER WILL RESPOND BY CHOOSING A CONTROLLER FROM THE"
1220 PRINT"SCHEMATIC AND TYPING IN ITS SYMBOL (ADDING *s AS NECESSARY TO"
1225 PRINT"MAKE 5-CHARACTER SYMBOLS)"
1230 PRINT
1240 PRINT"THUS, THE USER WILL TYPE IN:"
1250 PRINT
1260 PRINT"     TCA**"
1270 PRINT
1280 PRINT"THE COMPUTER WILL THEN PROCEED TO ASK FOR THE APPROPRIATE"
1290 PRINT"CONTROLLER PROGRAMMING INSTRUCTIONS AND ANY OTHER INFORMATION"
1300 PRINT"NEEDED TO RUN THE EXECUTION PROGRAM, SUCH AS THE INPUT COMPONENT"
1310 PRINT"TO THE CONTROLLER."
1320 PRINT
1330 PRINT"  2.  PRINT CONTENTS OF DATA DISK FILE"
1340 PRINT"BEFORE PROCEEDING, IT IS HIGHLY RECOMMENDED TO PRINT OUT THE"
1350 PRINT"CONTENTS OF THE DATA DISK FILE TO BE SURE THAT THE DATA FILE WAS"
1360 PRINT"IN FACT CREATED AND THE INFORMATION THAT WILL SUBSEQUENTLY BE FED"
1370 PRINT"TO THE EXECUTION PROGRAM IS ERROR FREE.  TO DO THIS, THE USER TYPES"
1380 PRINT"RUN ''CMODPRD'' (ie. DOUBLE QUOTES )."
1390 PRINT:PRINT
1400 INPUT "   TYPE IN ANY NUMBER, THEN HIT RETURN, TO CONTINUE";N
1410 ON N GOTO 1420
1420 PRINT
1430 PRINT"THE COMPUTER THEN ASKS FOR THE NAME OF THE DATA DISK FILE WHICH"
1440 PRINT"THE USED TYPES IN.  AS STATED EARLIER, THIS NAME IS UNIQUE, SUCH"
1450 PRINT"THAT EACH HVAC CONTROL SYSTEM MODELLED WILL BE REPRESENTED BY "
1460 PRINT"IT'S OWN DATA DISK FILE.  ONCE A DATA DISK FILE HAS BEEN CREATED,"
1470 PRINT"IT CAN BE SAVED, EXECUTED, OR MODIFIED AND REEXECUTED AT ANY TIME"
1480 PRINT"IN THE FUTURE."
1490 PRINT
1500 PRINT"WHEN CMODPRD IS RUN USING A DATA DISK FILE THAT HAS NOT YET BEEN"
1510 PRINT"EXECUTED (VIA CMODEXC), THE RESULT IS A PRINTOUT OF ALL COMPONENTS"
1520 PRINT"BY COMPONENT TYPE AND ALL INFORMATION NEEDED (PROGRAMMING INSTRUCT-"
1530 PRINT"IONS, INPUT COMPONENTS, ETC) TO MAKE THE COMPUTATIONS WHICH WILL"
1540 PRINT"PREDICT THE CONTROLS SYSTEM'S OUTPUTS.  THE OUTPUT VALUES WILL BE"
1550 PRINT"SHOWN AS 999 UNTIL THE EXECUTION PROGRAM HAS BEEN RUN."
1560 PRINT
1570 PRINT
1580 INPUT"   TYPE IN ANY NUMBER, THEN HIT RETURN, TO CONTINUE";N
1590 ON N GOTO 1600
1600 PRINT
1610 PRINT"  3.  MAKE COMPUTATIONS"
1620 PRINT"AT THIS POINT THE USER RUNS THE EXECUTION PRGORAM, CMODEXC."
1630 PRINT"CMODEXC USES AN ITERATIVE PROCESS AND HAS TWO MAIN FUNCTIONS."
1640 PRINT
1650 PRINT"FIRST, IT USES THE INFORMATION PROVIDED IN THE DATA DISK FILE"
1660 PRINT"CREATED IN CMODCRD AND PERFORMS ALL COMPUTATIONS NEEDED TO "
1670 PRINT"DETERMINE THE OUTPUT OF ALL COMPONENTS OF A HVAC CONTROL SYSTEM"
1680 PRINT"BEING MODELLED.  SECOND, IT UPDATES THE DATA DISK FILE SUCH THAT"
1690 PRINT"UPON TERMINATION, THE OUTPUT VALUE FOR THE SYSTEM AS WELL AS THE"
1700 PRINT"OUTPUTS OF ALL INTERMEDIATE COMPONENTS ARE CALCULATED AND UPDATED"
1710 PRINT"ON THE DATA DISK FILE."
1720 PRINT
1730 PRINT"CMODEXC EMPLOYS THE 'BRUTE FORCE METHOD', A WELL KNOWN TERM IN THE"
1740 PRINT"COMPUTER PROGRAMMING ARENA.  THE PROGRAM ITERATES AS MANY TIMES AS"
1750 PRINT"THERE ARE COMPONENTS IN THE HVAC CONTROL SYSTEM BEING MODELLED, AND"
1760 PRINT"EACH COMPONENT'S OUTPUT VALUE IS CALCULATED ONCE PER ITERATION."
1770 PRINT"NEVERTHELESS, CMOCEXC RUNS RATHER QUICKLY (APPROXIMATELY 2-3 "
1780 PRINT"SECONDS PER COMPONENT), SINCE CMODEXC DOES NOT CONSIDER "
1790 PRINT"TRANSIENT RESPONSES AND HIGH-LEVEL MATHEMATICS IS NOT USED."
1800 PRINT
1810 PRINT
1820 INPUT"   TYPE IN ANY NUMBER, THEN HIT RETURN, TO CONTINUE";N
1830 ON N GOTO 1840
1840 PRINT
1850 PRINT"  4.  REPRINT CONTENTS OF DATA DISK FILE"
1860 PRINT"THIS STEP IS NECESSARY TO OBTAIN A PRINTOUT OF THE REVISED DATA"
```

```
1870 PRINT"DISK FILE, WHICH INCLUDES ALL COMPONENT OUTPUT VALUES AS CALCULATED"
1880 PRINT"AND UPDATED IN CMODEXC."
1890 PRINT
1900 PRINT"  5.  EDIT DATA DISK FILE"
1910 PRINT"THE EDITING PROGRAM, CMODEDD, IS USED TO ALLOW THE USER TO CHANGE "
1920 PRINT"ANY OR ALL OF THE INPUT VALUES (SUCH AS A CONTROLLER SET POINT) ON"
1930 PRINT"AN EXISTING DATA DISK FILE.  THIS IS DONE IN ORDER TO BE ABLE TO"
1940 PRINT"PREDICT THE EFFECT THAT ANY SUCH CHANGES WOULD HAVE ON INDIVIDUAL"
1950 PRINT"CONTROLS COMPONENT OUTPUTS AND THE OVERALL CONTROL SYSTEM PERFORM-"
1960 PRINT"ANCE AS WELL."
1970 PRINT
1980 PRINT"IN PRACTICE, THE USER WOULD EDIT AN EXISTING DATA DISK FILE, RERUN"
1990 PRINT"CMODEXC USING THE EDITED DATA DISK FILE, AND THEN RERUN CMODPRD TO"
2000 PRINT"OBTAIN A NEW PRINTOUT.  THEN BY COMPARING AND CONTRASTING THE "
2010 PRINT"PRINTOUTS OF THE EDITED DATA DISK FILE TO THE ORIGINAL DATA DISK"
2020 PRINT"FILE, THE USER CAN OPTIMIZE EITHER A NEW OR EXISTING HVAC "
2030 PRINT"RECOMMEND IMPROVEMENTS TO AN EXISTING HVAC CONTROL SYSTEM."
2040 PRINT
2050 PRINT
2060 INPUT"  TYPE IN ANY NUMBER, THEN HIT RETURN, TO CONTINUE";N
2070 ON N GOTO 2080
2080 PRINT
2090 PRINT"DUE TO THE MODULARITY AND RUN-ALONE CAPABILITY OF THE CMOD"
2100 PRINT"PROGRAMS, THE EDIT/REEXECUTE/REPRINT CYCLE CAN BE REPEATED AS"
2110 PRINT"OFTEN AS THE USER DESIRES AT A FRACTION OF THE TIME REQUIRED TO"
2120 PRINT"DO THE SAME PROCESS MANUALLY."
2130 PRINT
2140 PRINT"TO GET A PRINTOUT OF CMODDOC"
2150 PRINT"****************************"
2160 PRINT"TO GET A PRINTOUT (HARD COPY) OF THESE INSTRUCTIONS, YOU MUST"
2170 PRINT"'LIST' THE ACTUAL PROGRAM.  TO DO THIS, TURN ON YOUR PRINTER"
2180 PRINT"AND TYPE:"
2190 PRINT
2200 PRINT"          LOAD ''CMODDOC'' "
2210 PRINT
2220 PRINT"(i.e., DOUBLE QUOTES AROUND CMODDOC)
2230 PRINT"THEN TYPE:"
2240 PRINT
2250 PRINT"          LLIST"
```

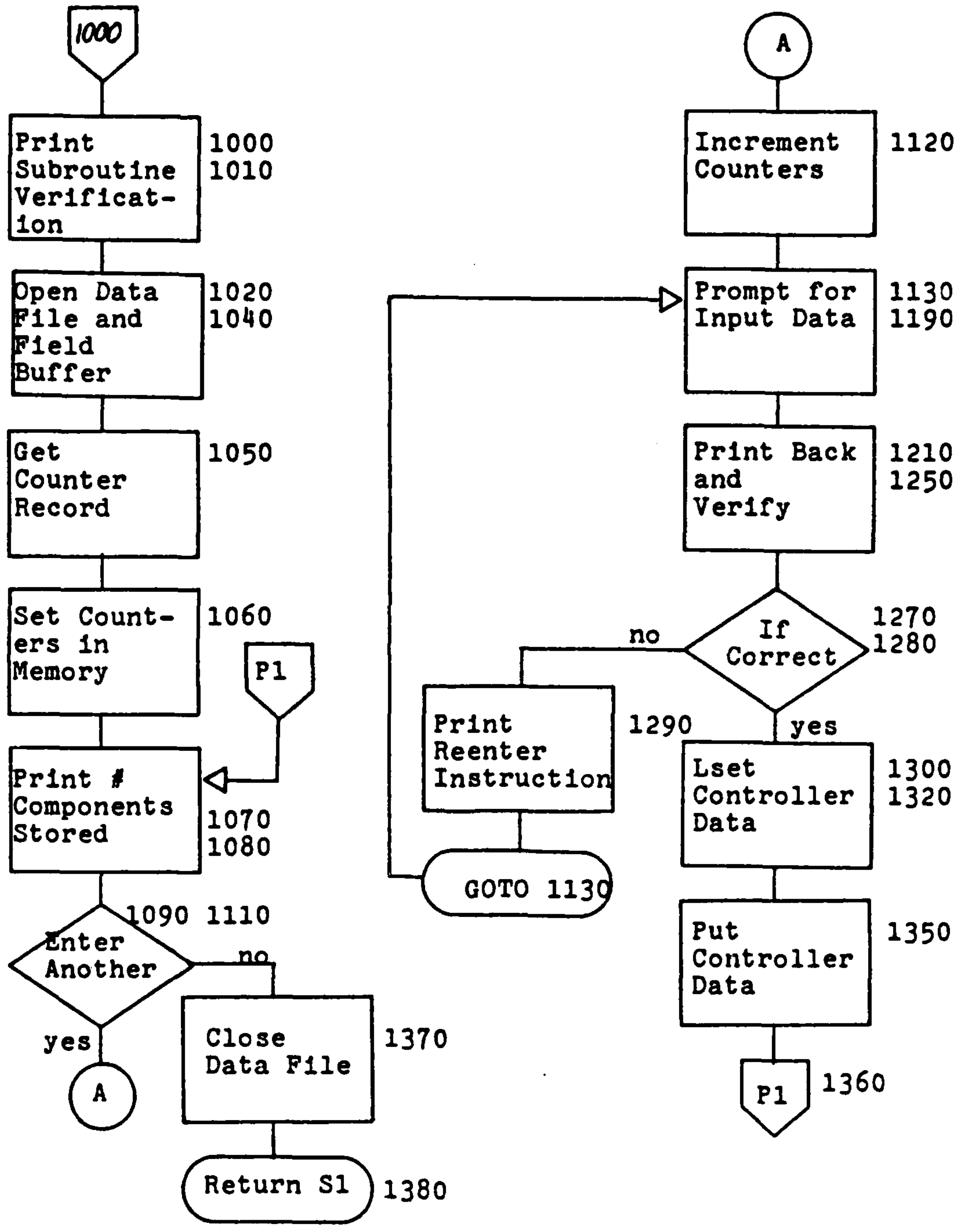
Appendix B. <u>Flowcharts of CMOD Programs</u>

Appendix B contains flow charts of the four working programs listed in Appendix A. The numbers which appear to the immediate right of the flow chart symbols refer to the program line numbers represented by the flow chart symbols. This feature facilitates easy cross-referencing between the computer programs and the flow charts.

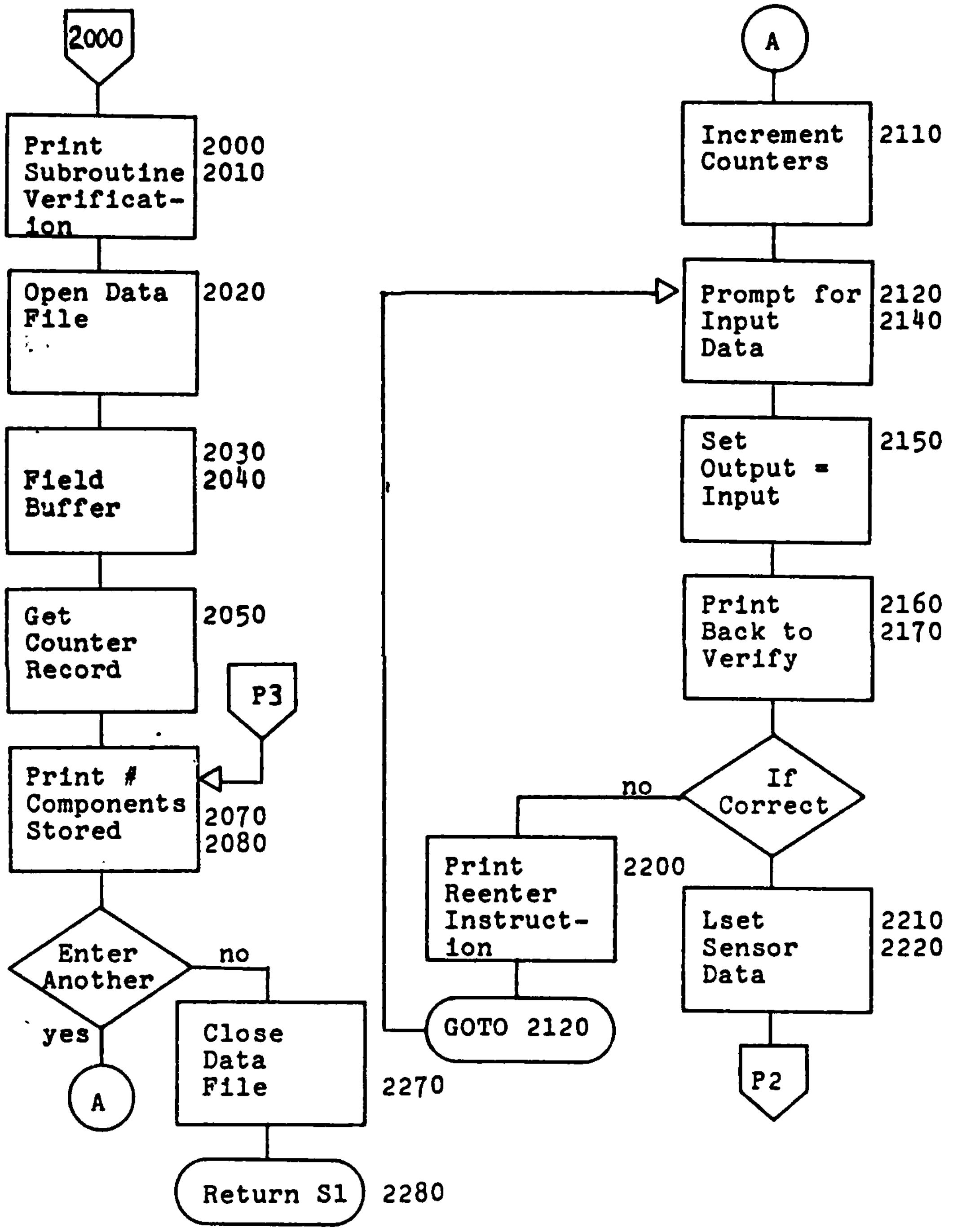
CMODCRD
Flow Chart



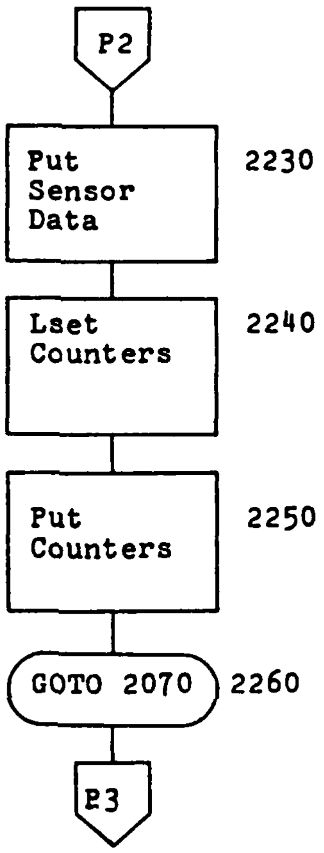
CMODCRD
Flow Chart Cont'd



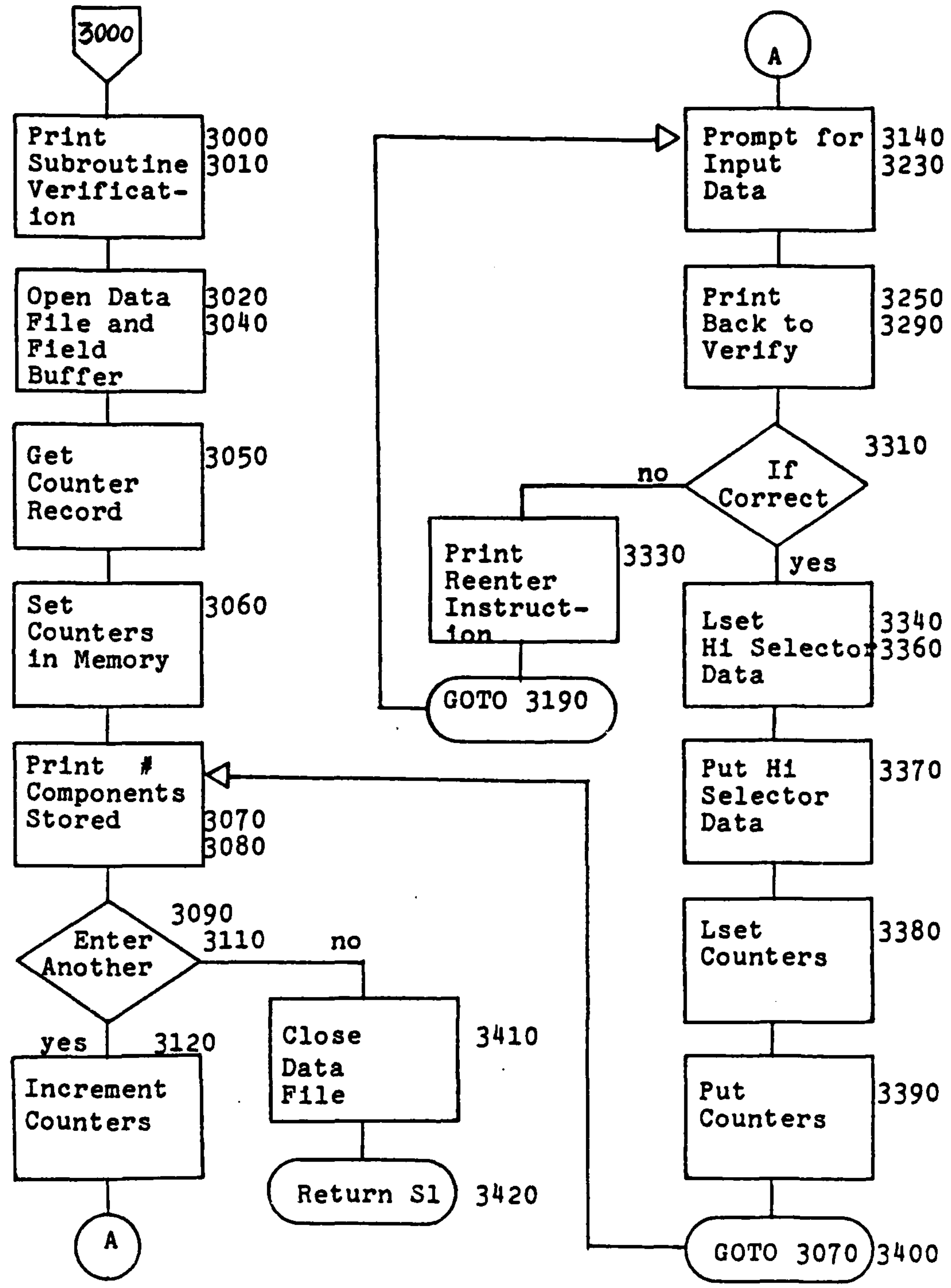
CMODCRD
Flow Chart Cont'd



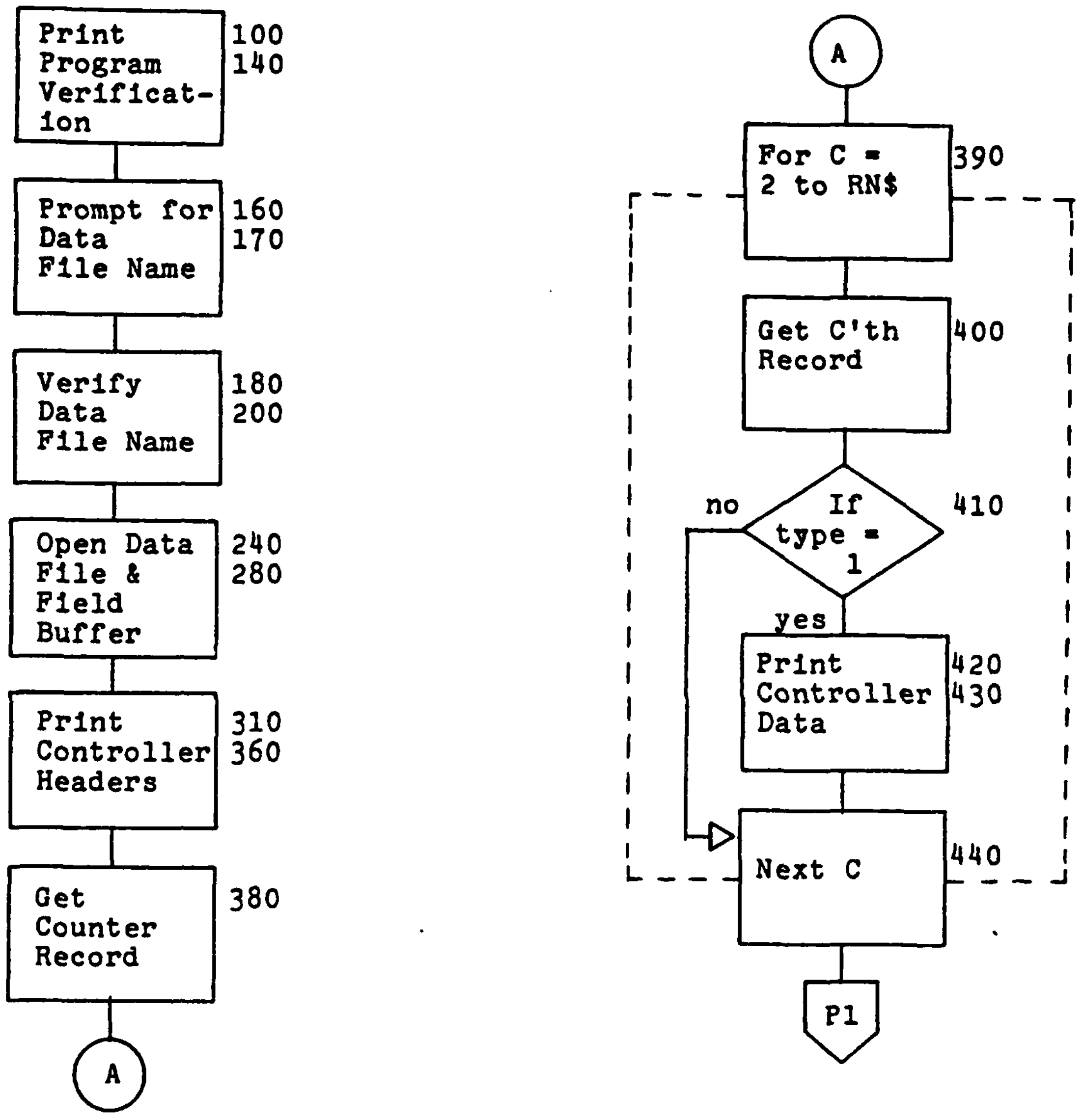
CMODCRD
Flow Chart Cont'd



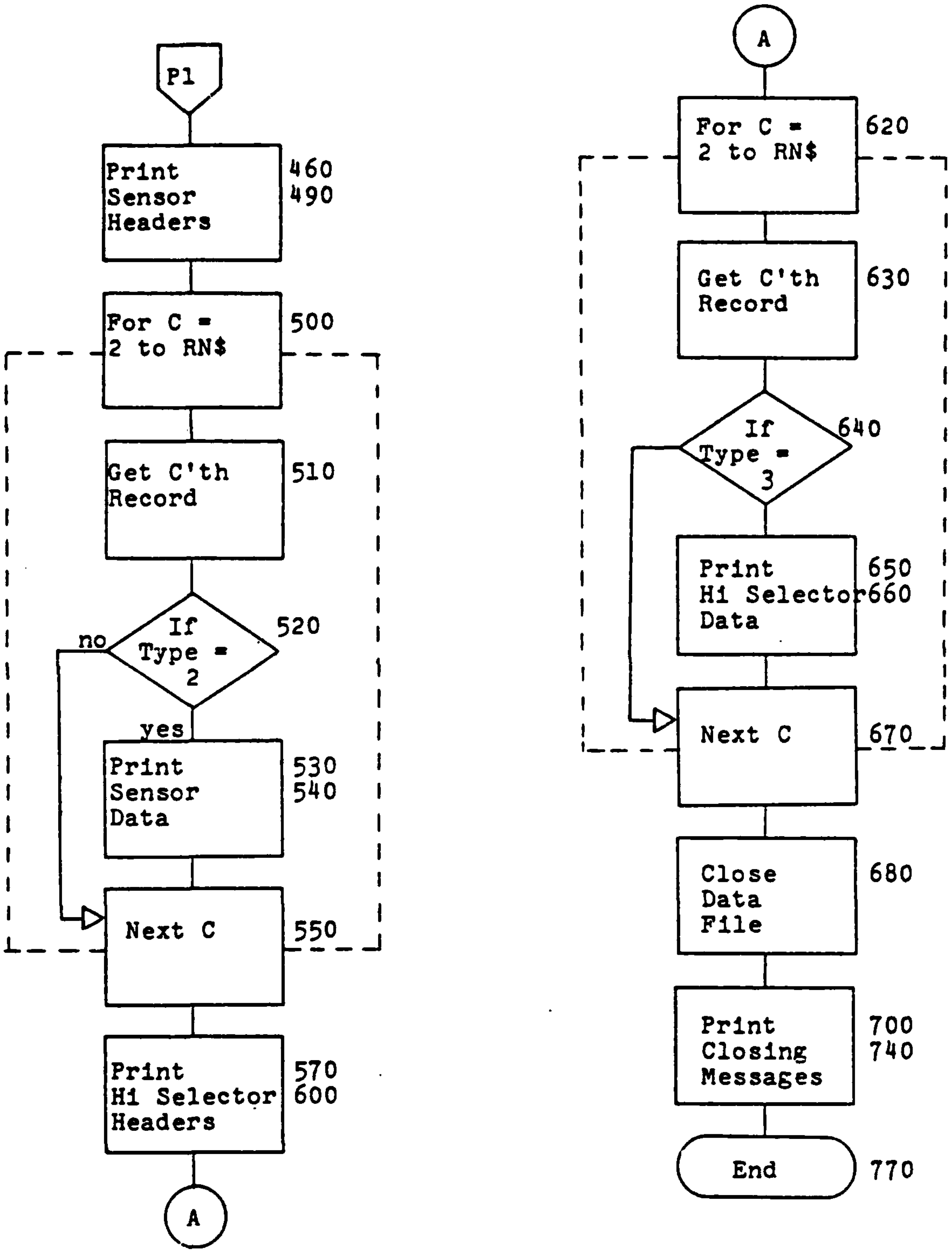
CMODCRD
Flow Chart Cont'd



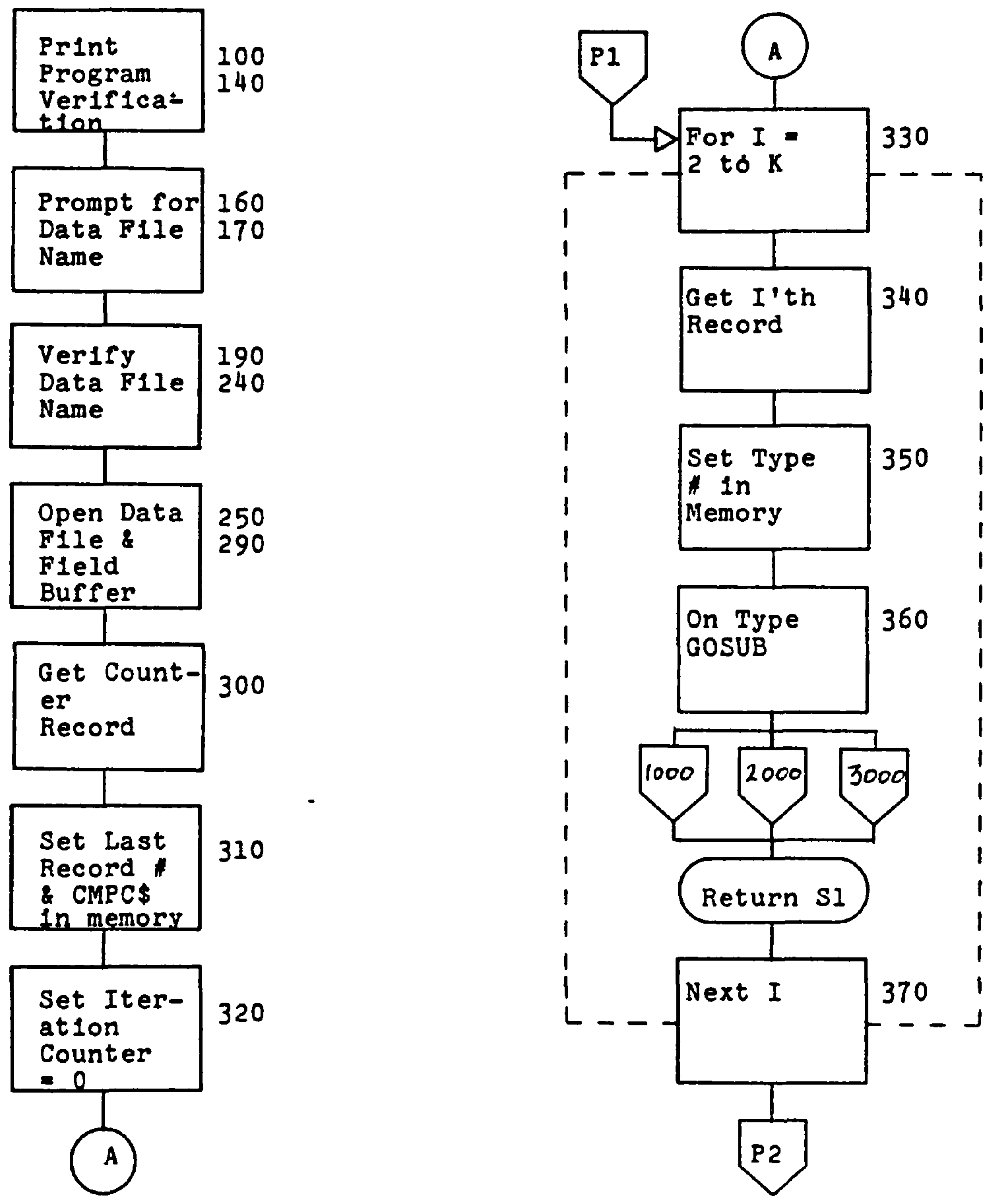
CMODPRD
Flow Chart



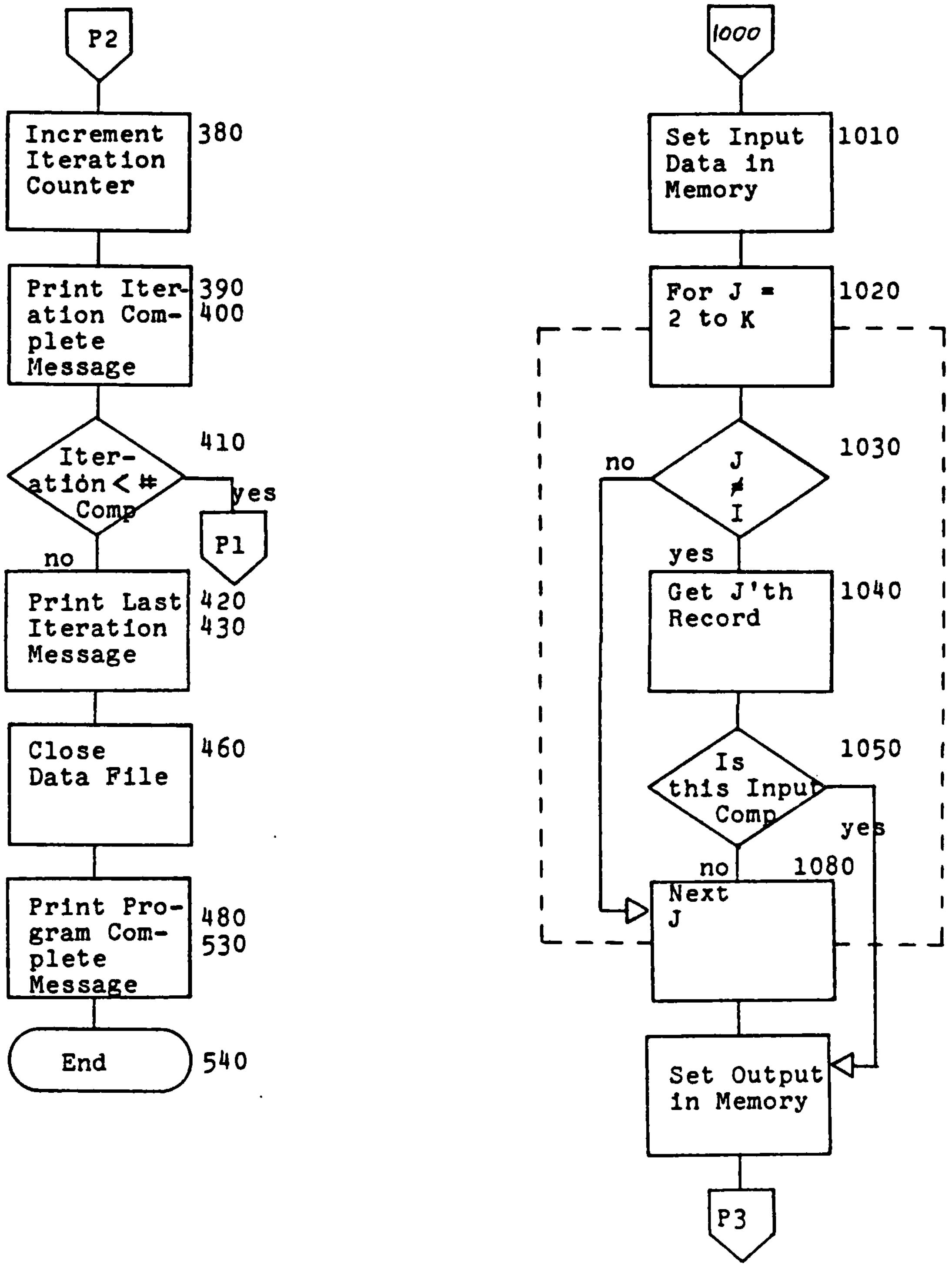
CMODPRD
Flow Chart Cont'd



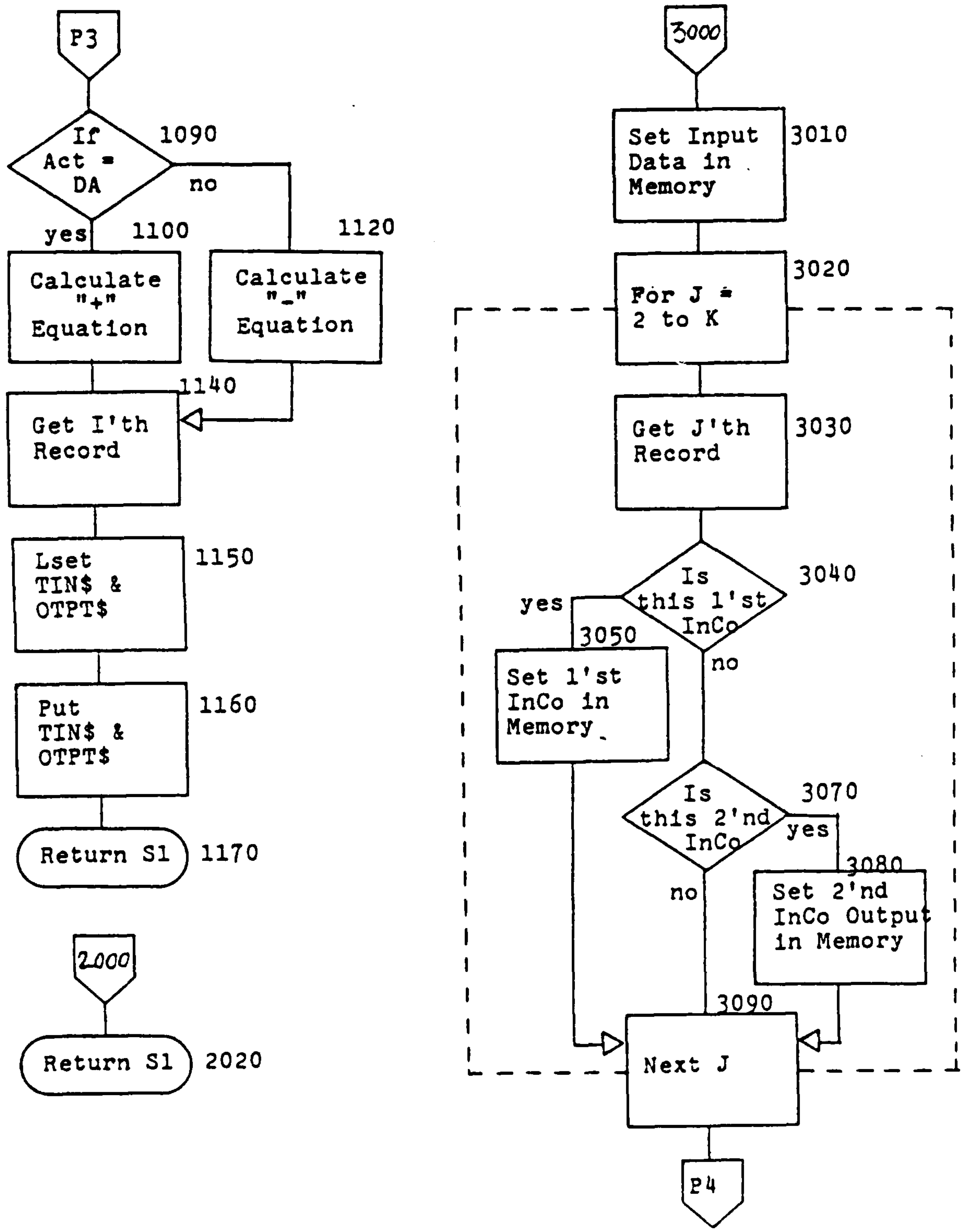
CMODEXC Flow Chart



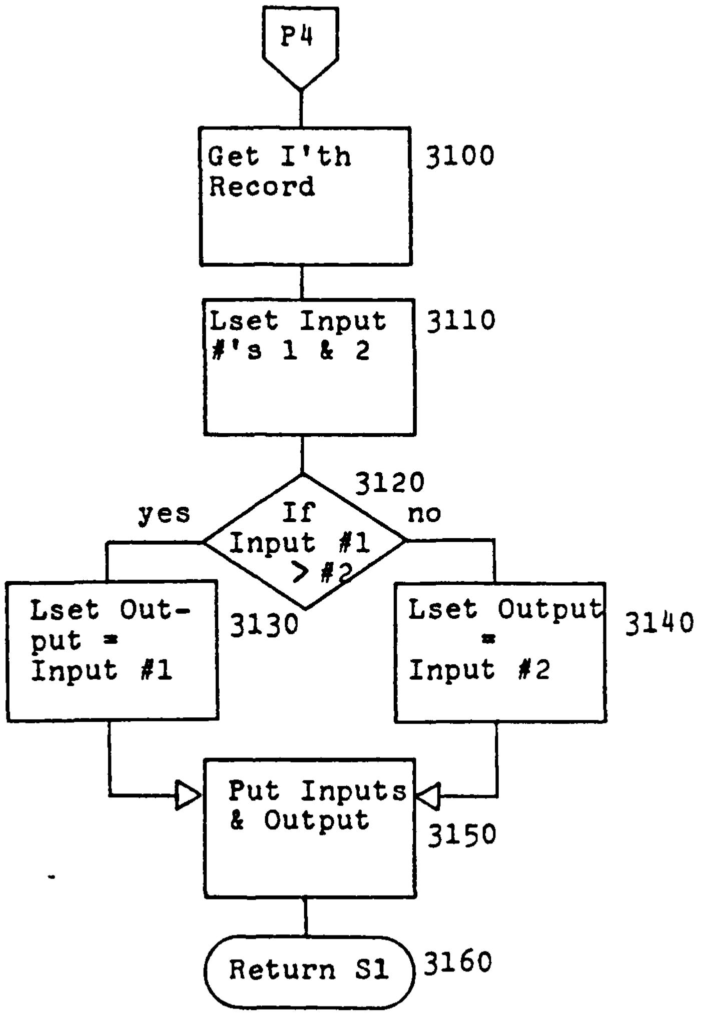
CMODEXC
Flow Chart Cont'd



CMODEXC
Flow Chart Cont'd

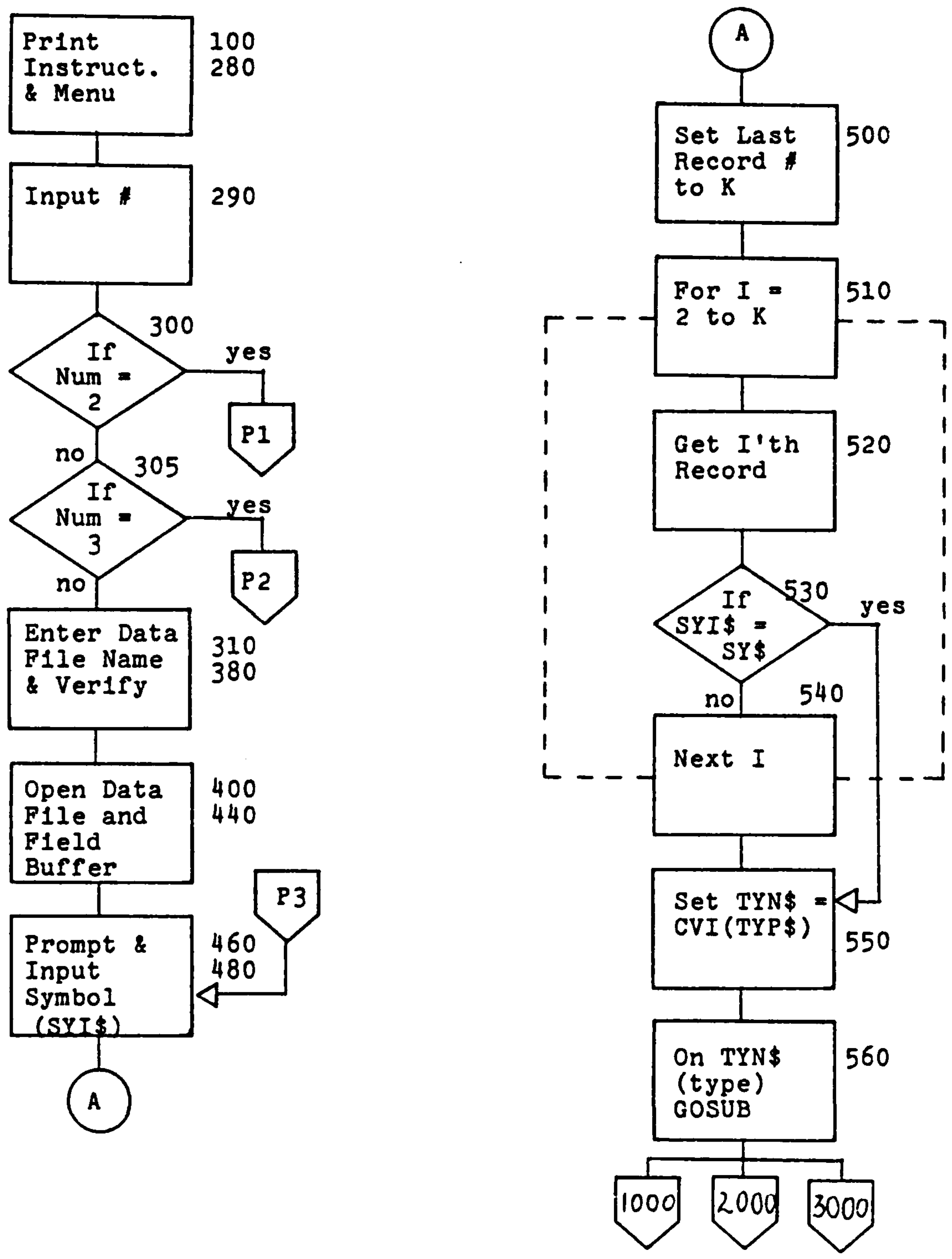


CMODEXC
Flow Chart Cont'd

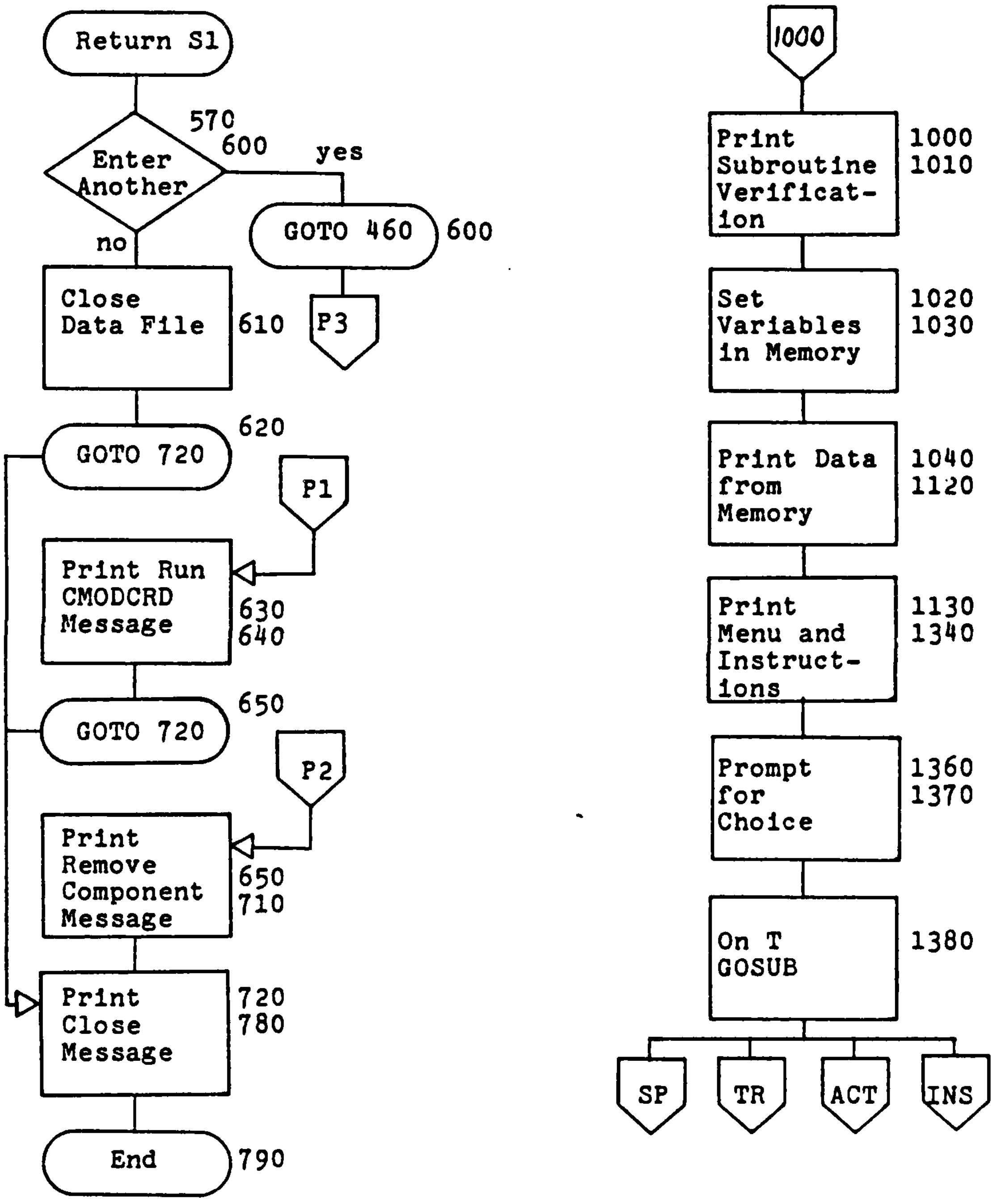


Note: I = Counter for Iteration
J = Counter for Input Component Search

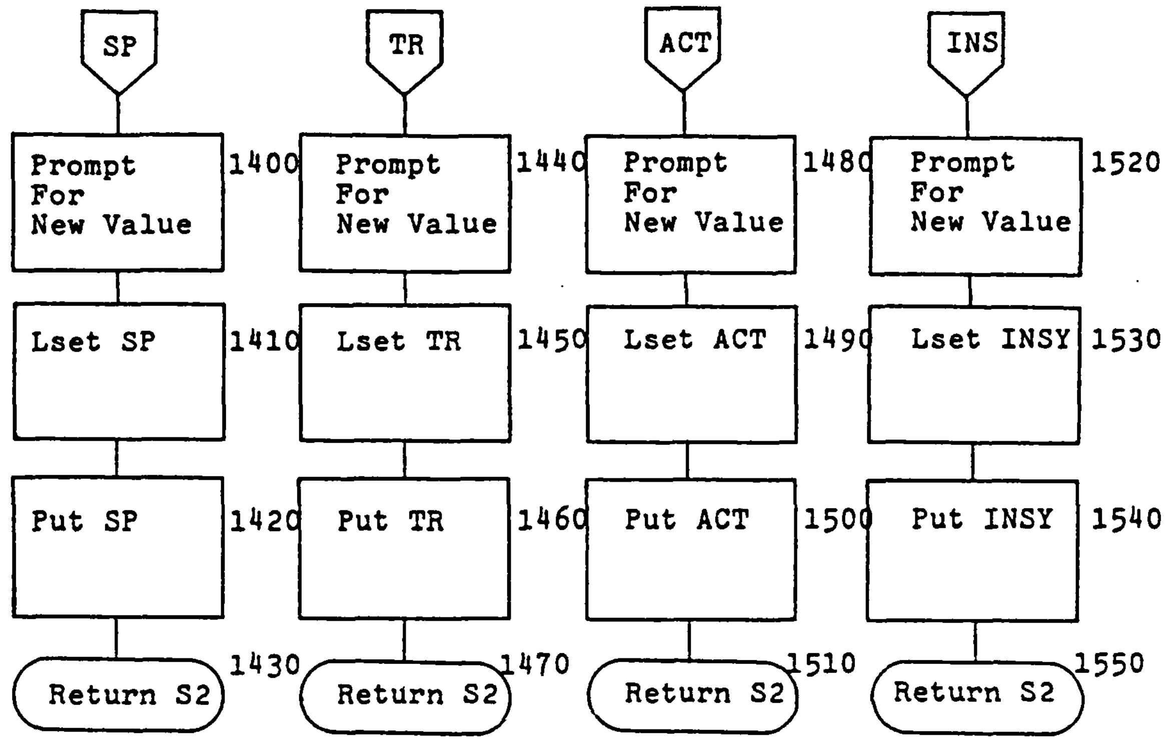
CMODEDD
Flow Chart

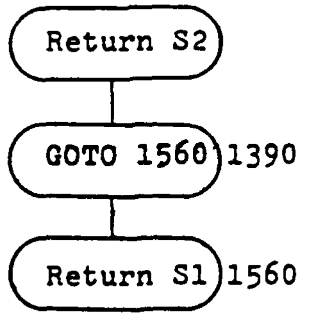


CMODEDD
Flow Chart Cont'd

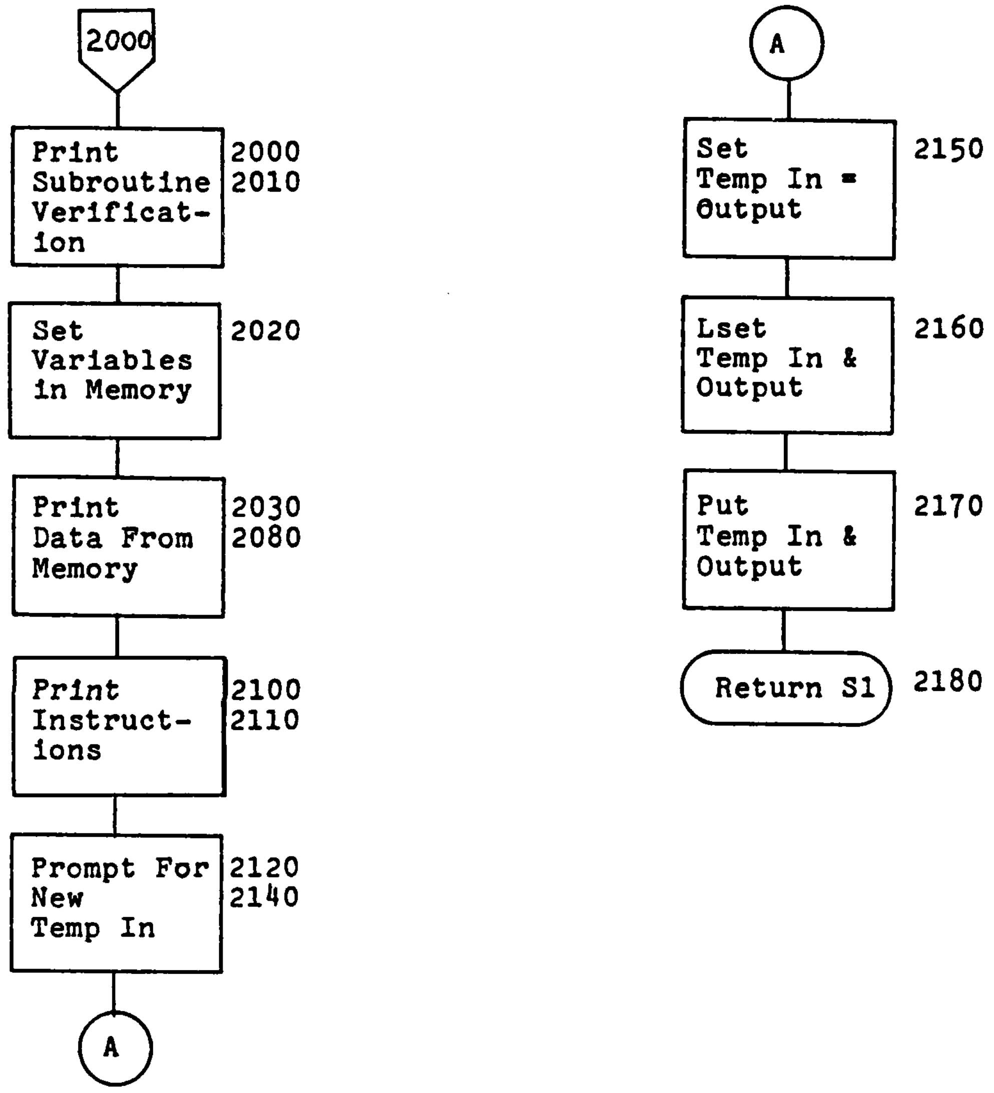


CMODEDD
Flow Chart Cont'd

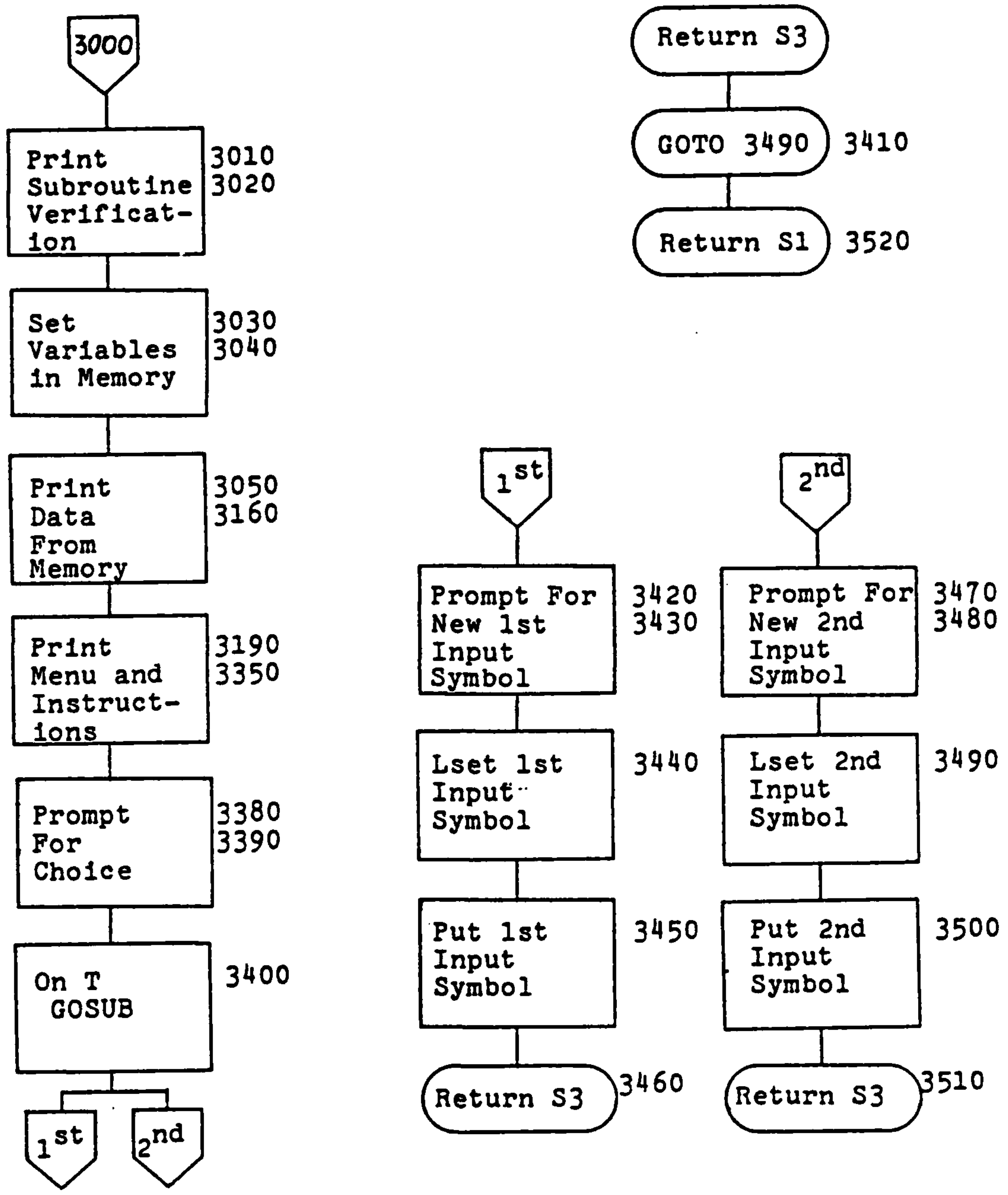




CMODEDD
Flow Chart Cont'd



CMODEDD
Flow Chart Cont'd



Appendix C. Sample Run of CMOD Programs

The eleven-component subsystem shown in Figure 1 of this thesis was modeled using the CMOD programs developed under this research effort. Appendix C contains the printouts that were produced in the modeling process. The handwritten text represents the responses typed in by the user during the "interactive" session while the typed text represents the information or prompts printed out via the CMOD programs.

Appendix C is subdivided into four "runs" in order to demonstrate the use of the CMOD programs and to show how the programs facilitate the optimization process described in the first two chapters of this thesis. Each run is described below.

The first run modeled the subsystem as it appears in Figure 1. First, a data disk file of subsystem was created using CMODCRD. Next, the contents of the data file were printed out using CMODPRD. Then the component output values were computed using CMODEXC and the results were printed out by running CMODPRD a second time.

In the second run, the input value of temperature sensor SA was changed from 70 to 74 degrees Farenheit using the program CMODEDD and the sensor edit subroutine. Next, the contents of the data file was printed out to insure the values for SA were properly edited. Finally, the execution program (CMODEXC) was run and the new output values were once again printed out using CMODPRD.

The CMODEDD hi signal selector edit subroutine was tested in the third run. Specifically, the first input value for HIA was changed from TCA to TCZ and the first input value for HIB was changed from TCZ to TCA. After these changes were made, the print-execute-print sequence was once again used.

In the fourth run, the CMODEDD controller edit subroutine was tested by changing the set point (SP) of TCB from 72 to 80 degrees fahrenheit and once again, using the print-execute-print sequence.

FIRST RUN

```
RUN"CMODCRD"
CREATE DATA FILE PROGRAM
************************


ENTER THE DATA FILE NAME (UP TO 8 LETTERS) FOR THIS SYSTEM CMODRUN

THE FILENAME ENTERED WAS CMODRUN

IS FILENAME CORRECT YES

IS CMODRUN A NEW OR EXISTING DATA FILENAME (NEW OR EXIST) NEW

COMPONENT MENU

1 = SINGLE INPUT CONTROLLER
2 = TEMPERATURE SENSOR
3 = HI SELECTOR

ENTER NUMBER FOR APPROPRIATE COMPONENT TYPE 1

SINGLE INPUT CONTROLLER DATA ENTRY SUBROUTINE

 0  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

ENTER ANOTHER CONTROLLER ? YES

(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)
(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)

CONTROLLER SYMBOL = ? TCA**
SET POINT = ? 71
THROT RANGE = ? 6
ACTION = (DA OR RA) ? DA
SYMBOL OF INPUT COMPONENT = SA***

FOR CONTROLLER TCA** -
SP =  71
TR =  6
ACT = DA
INPUT COMPONENT =SA***

IS THIS CORRECT ? YES
 1  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

ENTER ANOTHER CONTROLLER ? NO

DO YOU WANT TO ENTER ANOTHER COMPONENT (YES OR NO) YES
COMPONENT MENU

1 = SINGLE INPUT CONTROLLER
2 = TEMPERATURE SENSOR
3 = HI SELECTOR

ENTER NUMBER FOR APPROPRIATE COMPONENT TYPE 2

TEMP SENSOR DATA ENTRY SUBROUTINE

 1  COMPONENTS PRESENTLY STORED IN THIS DATA FILE
```

```
DO YOU WANT TO ENTER ANOTHER SENSOR (YES OR NO) YES

(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)
(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)

SENSOR SYMBOL = SA***
TEMP IN = 70

FOR SENSOR SA*** TEMP IN =  70
IS THIS CORRECT  YES


 2  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

DO YOU WANT TO ENTER ANOTHER SENSOR (YES OR NO) YES

(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)
(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)

SENSOR SYMBOL = SB***
TEMP IN = 72

FOR SENSOR SB*** TEMP IN =  72
IS THIS CORRECT  YES


 3  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

DO YOU WANT TO ENTER ANOTHER SENSOR (YES OR NO)  NO

DO YOU WANT TO ENTER ANOTHER COMPONENT (YES OR NO) YES
COMPONENT MENU

1 = SINGLE INPUT CONTROLLER
2 = TEMPERATURE SENSOR
3 = HI SELECTOR

ENTER NUMBER FOR APPROPRIATE COMPONENT TYPE  3

HI SELECTOR DATA ENTRY SUBROUTINE

 3  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

ENTER ANOTHER HI SELECTOR  YES

(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)
(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)

HI SELECTOR SYMBOL =  HIA
SYMBOL OF 1ST INPUT COMPONENT = TCA**
SYMBOL OF 2ND INPUT COMPONENT = TCB**

FOR HI SELECTOR HIA -
SYMBOL OF 1ST INPUT COMPONENT = TCA**
SYMBOL OF 2ND INPUT COMPONENT = TCB**

IS THIS CORRECT  NO
HI SELECTOR SYMBOL = HIA**
SYMBOL OF 1ST INPUT COMPONENT = TCA**
SYMBOL OF 2ND INPUT COMPONENT = TCB**

FOR HI SELECTOR HIA** -
SYMBOL OF 1ST INPUT COMPONENT = TCA**
SYMBOL OF 2ND INPUT COMPONENT = TCB**

IS THIS CORRECT YES
 4  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

ENTER ANOTHER HI SELECTOR YES
```

```
(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)
(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)

HI SELECTOR SYMBOL = HIC**
SYMBOL OF 1ST INPUT COMPONENT = HIA**
SYMBOL OF 2ND INPUT COMPONENT = HIB**

FOR HI SELECTOR HIC** -
SYMBOL OF 1ST INPUT COMPONENT = HIA**
SYMBOL OF 2ND INPUT COMPONENT = HIB**

IS THIS CORRECT YES
 5  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

ENTER ANOTHER HI SELECTOR NO

DO YOU WANT TO ENTER ANOTHER COMPONENT (YES OR NO) YES

COMPONENT MENU

1 = SINGLE INPUT CONTROLLER
2 = TEMPERATURE SENSOR
3 = HI SELECTOR

ENTER NUMBER FOR APPROPRIATE COMPONENT TYPE 1

SINGLE INPUT CONTROLLER DATA ENTRY SUBROUTINE

 5  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

ENTER ANOTHER CONTROLLER ? YES

(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)
(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)

CONTROLLER SYMBOL = ? TCB**
SET POINT = ? 72
THROT RANGE = ? 8
ACTION = (DA OR RA) ? RA
SYMBOL OF INPUT COMPONENT = SB***

FOR CONTROLLER TCB** -
SP =  72
TR =  8
ACT = RA
INPUT COMPONENT =SB***

IS THIS CORRECT ? YES
 6  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

ENTER ANOTHER CONTROLLER ? YES

(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)
(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)

CONTROLLER SYMBOL = ? TCZ**
SET POINT = ? 73
THROT RANGE = ? 10
ACTION = (DA OR RA) ? DA
SYMBOL OF INPUT COMPONENT = SC***

FOR CONTROLLER TCZ** -
SP =  73
TR =  10
ACT = DA
INPUT COMPONENT =SC***

IS THIS CORRECT ? YES
```

```
 7  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

ENTER ANOTHER CONTROLLER ?  YES

(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)
(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)

CONTROLLER SYMBOL = ? TCD**
SET POINT = ? 74
THROT RANGE = ? 10
ACTION = (DA OR RA) ? RA
SYMBOL OF INPUT COMPONENT = SD***

FOR CONTROLLER TCD** -
SP =  74
TR =  10
ACT = RA
INPUT COMPONENT =SD***

IS THIS CORRECT ? YES
 8  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

ENTER ANOTHER CONTROLLER ? NO

DO YOU WANT TO ENTER ANOTHER COMPONENT (YES OR NO)
COMPONENT MENU

1 = SINGLE INPUT CONTROLLER
2 = TEMPERATURE SENSOR
3 = HI SELECTOR

ENTER NUMBER FOR APPROPRIATE COMPONENT TYPE 2

TEMP SENSOR DATA ENTRY SUBROUTINE

 8  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

DO YOU WANT TO ENTER ANOTHER SENSOR (YES OR NO) YES

(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)
(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)

SENSOR SYMBOL = SD***
TEMP IN = 76

FOR SENSOR SD*** TEMP IN =  76
IS THIS CORRECT


 9  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

DO YOU WANT TO ENTER ANOTHER SENSOR (YES OR NO) YES

(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)
(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)

SENSOR SYMBOL = SC***
TEMP IN = 74

FOR SENSOR SC*** TEMP IN =  74
IS THIS CORRECT YES


 10  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

DO YOU WANT TO ENTER ANOTHER SENSOR (YES OR NO) NO

DO YOU WANT TO ENTER ANOTHER COMPONENT (YES OR NO) YES
COMPONENT MENU
```

```
1 = SINGLE INPUT CONTROLLER
2 = TEMPERATURE SENSOR
3 = HI SELECTOR

ENTER NUMBER FOR APPROPRIATE COMPONENT TYPE 3

HI SELECTOR DATA ENTRY SUBROUTINE

 10  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

ENTER ANOTHER HI SELECTOR YES

(FOR COMPONENT SYMBOLS, USE 5-CHARACTER SYMBOLS)
(ADD *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)

HI SELECTOR SYMBOL = HIB**
SYMBOL OF 1ST INPUT COMPONENT = TCZ**
SYMBOL OF 2ND INPUT COMPONENT = TCD**

FOR HI SELECTOR HIB** -
SYMBOL OF 1ST INPUT COMPONENT = TCZ**
SYMBOL OF 2ND INPUT COMPONENT = TCD**

IS THIS CORRECT YES
 11  COMPONENTS PRESENTLY STORED IN THIS DATA FILE

ENTER ANOTHER HI SELECTOR NO

DO YOU WANT TO ENTER ANOTHER COMPONENT (YES OR NO) NO

**CREATE** DATA FILE PROGRAM COMPLETE

SELECT ANOTHER PROGRAM BY TYPING 'RUN programname'


RUN "CMODPRD"


PRINT CONTENTS OF DATA FILE PROGRAM
**********************************

ENTER DATA FILE NAME (UP TO 8 LETTERS) FOR THIS SYSTEM CMODRUN
THE FILENAME ENTERED WAS CMODRUN
IS THIS CORRECT (YES OR NO) YES



DATA FOR CONTROLLERS

SYMBOL  ACTION  THROT RANGE  SET POINT  INPUT COMPONENT  TEMP IN  VOLT OUT

TCA**   DA      6             71          SA***           999       999
TCB**   RA      8             72          SB***           999       999
TCZ**   DA      10            73          SC***           999       999
TCD**   RA      10            74          SD***           999       999

DATA FOR SENSORS

SYMBOL  TEMP IN  OUTPUT

SA***    70         70
SB***    72         72
SD***    76         76
SC***    74         74
```

```
DATA FOR HI SELECTORS

SYMBOL  1ST INPUT SYMBOL   SYMBOL 1 IN   2ND INPUT SYMBOL   SYMBOL 2 IN  OUTPUT

HIA**      TCA**              999           TCB**              999        999
HIC**      HIA**              999           HIB**              999        999
HIB**      TCZ**              999           TCD**              999        999

**PRINT** PROGRAM RUN COMPLETE

SELECT ANOTHER PROGRAM BY TYPING RUN ''programname''(DOUBLE QUOTES)


RUN " CMODEXC

EXECUTION MODE PROGRAM
**********************

ENTER THE DATA FILE NAME (UP TO 8 LETTERS) CMODRUN

THE FILE NAME ENTERED WAS CMODRUN
IS FILE NAME CORRECT (YES OR NO) ? YES

 1  ITERATION(S) COMPLETE
 2  ITERATION(S) COMPLETE
 3  ITERATION(S) COMPLETE
 4  ITERATION(S) COMPLETE
 5  ITERATION(S) COMPLETE
 6  ITERATION(S) COMPLETE
 7  ITERATION(S) COMPLETE
 8  ITERATION(S) COMPLETE
 9  ITERATION(S) COMPLETE
 10  ITERATION(S) COMPLETE
 11  ITERATION(S) COMPLETE
FOR THIS  11  COMPONENT SYSTEM


**EXECUTION** PROGRAM RUN COMPLETE

RUN CMODPRD TO SEE RESULTS OF EXECUTION

RUN "CMODPRD"


PRINT CONTENTS OF DATA FILE PROGRAM
***********************************

ENTER DATA FILE NAME (UP TO 8 LETTERS) FOR THIS SYSTEM CMODRUN
THE FILENAME ENTERED WAS CMODRUN
IS THIS CORRECT (YES OR NO) YES



DATA FOR CONTROLLERS

SYMBOL  ACTION  THROT RANGE  SET POINT  INPUT COMPONENT  TEMP IN  VOLT OUT

TCA**   DA      6             71          SA***             70       7
TCB**   RA      8             72          SB***             72       7.5
TCZ**   DA      10             73          SC***             74       7.8
TCD**   RA      10             74          SD***             76       6.9
```

```
DATA FOR SENSORS

SYMBOL  TEMP IN   OUTPUT

SA***     70          70
SB***     72          72
SD***     76          76
SC***     74          74


DATA FOR HI SELECTORS

SYMBOL  1ST INPUT SYMBOL   SYMBOL 1 IN   2ND INPUT SYMBOL   SYMBOL 2 IN  OUTPUT

HIA**      TCA**                 7            TCB**                7.5       7.5
HIC**      HIA**                 7.5            HIB**                7.8       7.8
HIB**      TCZ**                 7.8            TCD**                6.9       7.8

**PRINT** PROGRAM RUN COMPLETE

SELECT ANOTHER PROGRAM BY TYPING RUN ''programname''(DOUBLE QUOTES)

RUN "CMODEDD"
```

SECOND RUN

```
DATA EDIT PROGRAM
*****************

SELECT OPTION BY TYPING IN THE NUMBER INDICATED ON MENU
                                MENU

               OPTION                                  NUMBER

EDIT CONTENTS OF EXISTING DATA FILE................. 1
ADD ADDITIONAL COMPONENTS TO EXISTING DATA FILE..... 2
REMOVE COMPONENTS FROM EXISTING DATA FILE........... 3

ENTER THE DATA FILE NAME (UP TO 8 LETTERS) CMODRUN
THE FILE NAME ENTERED WAS CMODRUN
IS FILE NAME CORRECT (YES OR NO) ?  YES

(REMEMBER TO USE *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)


TEMPERATURE SENSOR **EDIT** SUBROUTINE

THE DATA PRESENTLY STORED FOR SA*** IS

TEMP IN =  70
OUTPUT =  70

THE ONLY ITEM YOU CAN CHANGE IS 'TEMP IN'

NEW VALUE FOR TEMP IN = 74

DO YOU WANT TO EDIT ANBOTHER COMPONENT  (YES OR NO) ?


**EDIT** PROGRAM RUN COMPLETE

RUN CMODEXC TO UPDATE DATA FILE TO REFLECT NEW OUTPUTS
```

```
PRINT CONTENTS OF DATA FILE PROGRAM
***********************************

ENTER DATA FILE NAME (UP TO 8 LETTERS) FOR THIS SYSTEM CMODRUN
THE FILENAME ENTERED WAS CMODRUN
IS THIS CORRECT (YES OR NO) YES



DATA FOR CONTROLLERS

SYMBOL  ACTION  THROT RANGE  SET POINT  INPUT COMPONENT  TEMP IN  VOLT OUT

TCA**    DA       6             71          SA***          70       7
TCB**    RA       8             72          SB***          72       7.5
TCZ**    DA       10             73          SC***          74       7.8
TCD**    RA       10             74          SD***          76       6.9

DATA FOR SENSORS

SYMBOL  TEMP IN  OUTPUT

SA***     74        74
SB***     72        72
SD***     76        76
SC***     74        74


DATA FOR HI SELECTORS

SYMBOL  1ST INPUT SYMBOL   SYMBOL 1 IN   2ND INPUT SYMBOL   SYMBOL 2 IN  OUTPUT

HIA**      TCA**              7             TCB**               7.5       7.5
HIC**      HIA**              7.5             HIB**              7.8       7.8
HIB**      TCZ**              7.8             TCD**              6.9       7.8

**PRINT** PROGRAM RUN COMPLETE

SELECT ANOTHER PROGRAM BY TYPING RUN ''programname''(DOUBLE QUOTES)

RUN "CMODEXC"
EXECUTION MODE PROGRAM
**********************

ENTER THE DATA FILE NAME (UP TO 8 LETTERS)

THE FILE NAME ENTERED WAS CMODRUN
IS FILE NAME CORRECT (YES OR NO) ?

 1  ITERATION(S) COMPLETE
 2  ITERATION(S) COMPLETE
 3  ITERATION(S) COMPLETE
 4  ITERATION(S) COMPLETE
 5  ITERATION(S) COMPLETE
 6  ITERATION(S) COMPLETE
 7  ITERATION(S) COMPLETE
 8  ITERATION(S) COMPLETE
 9  ITERATION(S) COMPLETE
 10  ITERATION(S) COMPLETE
 11  ITERATION(S) COMPLETE
FOR THIS  11  COMPONENT SYSTEM


**EXECUTION** PROGRAM RUN COMPLETE

RUN CMODPRD TO SEE RESULTS OF EXECUTION
```

```
PRINT CONTENTS OF DATA FILE PROGRAM
**********************************

ENTER DATA FILE NAME (UP TO 8 LETTERS) FOR THIS SYSTEM CMODRUN
THE FILENAME ENTERED WAS CMODRUN
IS THIS CORRECT (YES OR NO) YES



DATA FOR CONTROLLERS

SYMBOL  ACTION  THROT RANGE  SET POINT  INPUT COMPONENT  TEMP IN  VOLT OUT

TCA**   DA      6             71         SA***            74       9
TCB**   RA      8             72         SB***            72       7.5
TCZ**   DA      10            73          SC***            74       7.8
TCD**   RA      10            74          SD***            76       6.9

DATA FOR SENSORS

SYMBOL  TEMP IN  OUTPUT

SA***    74        74
SB***    72        72
SD***    76        76
SC***    74        74

DATA FOR HI SELECTORS

SYMBOL  1ST INPUT SYMBOL  SYMBOL 1 IN  2ND INPUT SYMBOL  SYMBOL 2 IN  OUTPUT

HIA**     TCA**             9            TCB**              7.5        9
HIC**     HIA**             9            HIB**              7.8        9
  HIB**     TCZ**             7.8          TCD**              6.9        7.8

**PRINT** PROGRAM RUN COMPLETE

SELECT ANOTHER PROGRAM BY TYPING RUN ''programname''(DOUBLE QUOTES)


 RUN "CMODEDD"
```

THIRD RUN

```
DATA EDIT PROGRAM
*****************

SELECT OPTION BY TYPING IN THE NUMBER INDICATED ON MENU
                              MENU

               OPTION                              NUMBER

EDIT CONTENTS OF EXISTING DATA FILE................. 1
ADD ADDITIONAL COMPONENTS TO EXISTING DATA FILE..... 2
REMOVE COMPONENTS FROM EXISTING DATA FILE........... 3

ENTER THE DATA FILE NAME (UP TO 8 LETTERS) CMODRUN
THE FILE NAME ENTERED WAS CMODRUN
IS FILE NAME CORRECT (YES OR NO) ? YES

(REMEMBER TO USE *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)
```

```
HI SELECTOR **EDIT** SUBROUTINE

THE DATA PRESENTLY STORED FOR HIA** IS

1ST INPUT COMPONENT = TCA**
INPUT SIGNAL FROM TCA** =  9
2ND INPUT COMPONENT = TCB**
INPUT SIGNAL FROM TCB** =  7.5
OUTPUT SIGNAL =  9


SELECT ITEM YOU WISH TO CHANGE FROM MENU

                       MENU
   TO CHANGE :                  TYPE IN :
1ST INPUT COMPONENT SYMBOL         1
2ND INPUT COMPONENT SYMBOL         2

TO CHANGE THE VALUE OF TCA** YOU MUST 'EDIT' TCA**

TO CHANGE THE VALUE OF TCB** YOU MUST 'EDIT' TCB**


TYPE IN APPROPRIATE NUMBER 1
NEW SYMBOL OF 1ST INPUT COMPONENT = TCZ**

DO YOU WANT TO EDIT ANBOTHER COMPONENT  (YES OR NO) ? YES

(REMEMBER TO USE *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)

HI SELECTOR **EDIT** SUBROUTINE

THE DATA PRESENTLY STORED FOR HIB** IS

1ST INPUT COMPONENT = TCA**
INPUT SIGNAL FROM TCZ** =  7.8
2ND INPUT COMPONENT = TCD**
INPUT SIGNAL FROM TCD** =  6.9
OUTPUT SIGNAL =  7.8


SELECT ITEM YOU WISH TO CHANGE FROM MENU

                       MENU
   TO CHANGE :                  TYPE IN :
1ST INPUT COMPONENT SYMBOL         1
2ND INPUT COMPONENT SYMBOL         2

TO CHANGE THE VALUE OF TCZ** YOU MUST 'EDIT' TCZ**

TO CHANGE THE VALUE OF TCD** YOU MUST 'EDIT' TCD**


TYPE IN APPROPRIATE NUMBER 1
NEW SYMBOL OF 1ST INPUT COMPONENT = TCA**

DO YOU WANT TO EDIT ANBOTHER COMPONENT  (YES OR NO) ? No


**EDIT** PROGRAM RUN COMPLETE

RUN CMODEXC TO UPDATE DATA FILE TO REFLECT NEW OUTPUTS
```

```
PRINT CONTENTS OF DATA FILE PROGRAM
***********************************

ENTER DATA FILE NAME (UP TO 8 LETTERS) FOR THIS SYSTEM CMODRUN
THE FILENAME ENTERED WAS CMODRUN
IS THIS CORRECT (YES OR NO) YES



DATA FOR CONTROLLERS

SYMBOL   ACTION   THROT RANGE   SET POINT   INPUT COMPONENT   TEMP IN   VOLT OUT

TCA**    DA       6              71            SA***             74        9
TCB**    RA       8              72            SB***             72        7.5
TCZ**    DA       10              73            SC***             74         7.8
TCD**    RA       10              74            SD***             76         6.9

DATA FOR SENSORS

SYMBOL   TEMP IN   OUTPUT

SA***     74          74
SB***     72          72
SD***     76          76
SC***     74          74

DATA FOR HI SELECTORS

SYMBOL   1ST INPUT SYMBOL   SYMBOL 1 IN   2ND INPUT SYMBOL   SYMBOL 2 IN   OUTPUT

HIA**      TCZ**                9            TCB**              7.5        9
HIC**      HIA**                9            HIB**              7.8        9
  HIB**      TCA**                7.8            TCD**              6.9        7.8

**PRINT** PROGRAM RUN COMPLETE

SELECT ANOTHER PROGRAM BY TYPING RUN ''programname''(DOUBLE QUOTES)


RUN "CMODEXC"


EXECUTION MODE PROGRAM
**********************

ENTER THE DATA FILE NAME (UP TO 8 LETTERS) CMODRUN

THE FILE NAME ENTERED WAS CMODRUN
IS FILE NAME CORRECT (YES OR NO) ? YES

 1   ITERATION(S) COMPLETE
 2   ITERATION(S) COMPLETE
 3   ITERATION(S) COMPLETE
 4   ITERATION(S) COMPLETE
 5   ITERATION(S) COMPLETE
 6   ITERATION(S) COMPLETE
 7   ITERATION(S) COMPLETE
 8   ITERATION(S) COMPLETE
 9   ITERATION(S) COMPLETE
 10   ITERATION(S) COMPLETE
 11   ITERATION(S) COMPLETE
FOR THIS  11  COMPONENT SYSTEM
```

```
**EXECUTION** PROGRAM RUN COMPLETE

RUN CMODPRD TO SEE RESULTS OF EXECUTION




RUN "CMODPRD"

PRINT CONTENTS OF DATA FILE PROGRAM
**********************************

ENTER DATA FILE NAME (UP TO 8 LETTERS) FOR THIS SYSTEM  CMODRUN
THE FILENAME ENTERED WAS CMODRUN
IS THIS CORRECT (YES OR NO) YES



DATA FOR CONTROLLERS

SYMBOL  ACTION  THROT RANGE  SET POINT  INPUT COMPONENT  TEMP IN  VOLT OUT

TCA**    DA       6             71          SA***            74       9
TCB**    RA       8             72          SB***            72       7.5
TCZ**    DA       10             73          SC***            74       7.8
TCD**    RA       10             74          SD***            76       6.9

DATA FOR SENSORS

SYMBOL  TEMP IN  OUTPUT

SA***    74         74
SB***    72         72
SD***    76         76
SC***    74         74

DATA FOR HI SELECTORS

SYMBOL  1ST INPUT SYMBOL   SYMBOL 1 IN   2ND INPUT SYMBOL   SYMBOL 2 IN  OUTPUT

HIA**     TCZ**                7.8          TCB**               7.5        7.8
HIC**     HIA**                7.8          HIB**               9        9
HIB**     TCA**                9          TCD**               6.9        9

**PRINT** PROGRAM RUN COMPLETE

SELECT ANOTHER PROGRAM BY TYPING RUN ''programname''(DOUBLE QUOTES)


RUN" CMODEDD"
```

FOURTH RUN

```
DATA EDIT PROGRAM
*****************

SELECT OPTION BY TYPING IN THE NUMBER INDICATED ON MENU
                               MENU

              OPTION                               NUMBER

EDIT CONTENTS OF EXISTING DATA FILE................. 1
ADD ADDITIONAL COMPONENTS TO EXISTING DATA FILE..... 2
REMOVE COMPONENTS FROM EXISTING DATA FILE........... 3

ENTER THE DATA FILE NAME (UP TO 8 LETTERS) CMODRUN
THE FILE NAME ENTERED WAS CMODRUN
IS FILE NAME CORRECT (YES OR NO) ? YES

(REMEMBER TO USE *s AS NEEDED TO GET 5-CHARACTER SYMBOLS)


SINGLE INPUT CONTROLLER **EDIT** SUBROUTINE

THE DATA PRESENTLY STORED FOR TCB** IS

SP =  72
TR =  8
ACT = RA
INPUT COMPONENT = SB***
TEMP IN =  72
OUTPUT =  7.5


SELECT ITEM YOU WISH TO CHANGE FROM MENU

                        MENU
TO CHANGE :                    TYPE IN :
  SET POINT                        1
  THROT RANGE                      2
  ACTION                           3
  INPUT SYMBOL                     4

TO CHANGE THE VALUE OF 'TEMP IN', YOU MUST 'EDIT' SB***

TYPE IN APPROPRIATE NUMBER 1
NEW SET POINT = 80

DO YOU WANT TO EDIT ANBOTHER COMPONENT  (YES OR NO) ? NO


**EDIT** PROGRAM RUN COMPLETE

RUN CMODEXC TO UPDATE DATA FILE TO REFLECT NEW OUTPUTS


RUN"CMODPRD"
```

```
PRINT CONTENTS OF DATA FILE PROGRAM
***********************************

ENTER DATA FILE NAME (UP TO 8 LETTERS) FOR THIS SYSTEM
THE FILENAME ENTERED WAS CMODRUN
IS THIS CORRECT (YES OR NO)



DATA FOR CONTROLLERS

SYMBOL  ACTION   THROT RANGE   SET POINT   INPUT COMPONENT   TEMP IN   VOLT OUT

TCA**    DA       6              71          SA***           74        9
TCB**    RA       8              80          SB***           72        7.5
TCZ**    DA       10              73          SC***           74        7.8
TCD**    RA       10              74          SD***           76        6.9

DATA FOR SENSORS

SYMBOL  TEMP IN  OUTPUT

SA***    74         74
SB***    72         72
SD***    76         76
SC***    74         74

DATA FOR HI SELECTORS

SYMBOL  1ST INPUT SYMBOL   SYMBOL 1 IN   2ND INPUT SYMBOL   SYMBOL 2 IN  OUTPUT

HIA**      TCZ**               7.8            TCB**            7.5       7.8
HIC**      HIA**               7.8            HIB**            9         9
HIB**      TCA**               9            TCD**            6.9         9

**PRINT** PROGRAM RUN COMPLETE

SELECT ANOTHER PROGRAM BY TYPING RUN ''programname''(DOUBLE QUOTES)


RUN "CMODEXC"

EXECUTION MODE PROGRAM
**********************

ENTER THE DATA FILE NAME (UP TO 8 LETTERS) CMODRUN

THE FILE NAME ENTERED WAS CMODRUN
IS FILE NAME CORRECT (YES OR NO) ? YES

 1  ITERATION(S) COMPLETE
 2  ITERATION(S) COMPLETE
 3  ITERATION(S) COMPLETE
 4  ITERATION(S) COMPLETE
 5  ITERATION(S) COMPLETE
 6  ITERATION(S) COMPLETE
 7  ITERATION(S) COMPLETE
 8  ITERATION(S) COMPLETE
 9  ITERATION(S) COMPLETE
 10  ITERATION(S) COMPLETE
 11  ITERATION(S) COMPLETE
FOR THIS  11  COMPONENT SYSTEM
```

```
**EXECUTION** PROGRAM RUN COMPLETE

RUN CMODPRD TO SEE RESULTS OF EXECUTION




RUN "CMODPRD"

PRINT CONTENTS OF DATA FILE PROGRAM
***********************************

ENTER DATA FILE NAME (UP TO 8 LETTERS) FOR THIS SYSTEM CMODRUN
THE FILENAME ENTERED WAS CMODRUN
IS THIS CORRECT (YES OR NO) YES



DATA FOR CONTROLLERS

SYMBOL  ACTION  THROT RANGE  SET POINT  INPUT COMPONENT  TEMP IN  VOLT OUT

TCA**   DA      6              71          SA***           74        9
TCB**   RA      8              80          SB***           72        10.5
TCZ**   DA      10              73          SC***           74        7.8
TCD**   RA      10              74          SD***           76        6.9

DATA FOR SENSORS

SYMBOL  TEMP IN  OUTPUT

SA***    74        74
SB***    72        72
SD***    76        76
SC***    74        74

DATA FOR HI SELECTORS

SYMBOL  1ST INPUT SYMBOL   SYMBOL 1 IN   2ND INPUT SYMBOL   SYMBOL 2 IN  OUTPUT

HIA**      TCZ**                 7.8           TCB**                10.5       10.5

HIC**      HIA**                10.5            HIB**                 9       10.5
HIB**      TCA**                9            TCD**              6.9        9

**PRINT** PROGRAM RUN COMPLETE

SELECT ANOTHER PROGRAM BY TYPING RUN ''programname''(DOUBLE QUOTES)
```

Bibliography

1. Building Energy Analysis Users Manual. Users Manual. Elite Software Development Inc., Bryan, 1983.

2. Clark, D. R., Hurley, C. W., and Hill, C. R. Ph.D. "Dynamic Models for HVAC System Components," ASHRAE Transactions, 91 1B: 737-751 (1985).

3. Department of the Air Force. Engineering Technical Letter 83-1: Design of Control Systems for Heating, Ventilating and Air Conditioning Systems (HVAC). ETL 83-1. Washington: HQ USAF, 16 February 1983.

4. Department of the Air Force. Heating, Ventilating, and Air Conditioning Systems: OPERATIONS AND MAINTENANCE. AFR 91-39 Draft. Washington: HQ USAF, Not yet released.

5. Department of the Air Force. Direct Digital Control of Heating, Ventilating, and Air Conditioning Equipment. Policy Letter. Tyndall AFB: HQ AFESC, 21 March 1985.

6. Direct Digital Control of Heating, Ventilating, & Air Conditioning Stystem. Brochure. Computer Controls Corporation, Wilmington MA, undated.

7. Gottfried, Byron S. Schaum's Outline Series of Theory and Problems: Programming with BASIC. New York: McGraw Hill Book Company, 1982.

8. Hackner, R. J., Mitchell, P. E. Ph.D, and Beckman, W. A. Ph.D., "HVAC Systems Dynamics and Energy Use in Buildings--Part II," ASHRAE Transactions, 91 1B: 781-795 (1985).

9. Haines, R. W. Control Systems for Heating, Ventilating, and Air Conditioning (Second Edition). Albany: Delmar Publishers, 1978.

10. Hill, C. R. Ph.D. "Simulation of a Multizone Air Handler." ASHRAE Transactions, 91 1B: 752-765 (1985).

11. HVAC Basics: Fundamentals of Control. Company Training Manual. Johnson Controls, Inc. Milwaukee WI, 1980 edition.

12. Jakobczyr, Jack S. "Direct Digital Control for VAV

Terminals," Heating Piping and Air Conditioning, 54 2: 77-81 (February 1982).

13. Lau, A. S et al. "Development of Computerized Control Strategies for a Large Chilled Water Plant," ASHRAE Transactions, 91 1B: 766-780 (1985).

14. Mau, Ernest E. Secrets of Better Basic. Rochelle Park NJ: Hayden Book Co., 1983.

15. Research Report
Schultz, Maj Robert J., Kenna, Maj Thomas M. and Kapka, Capt Larraine A. An Investigation into the Operation and Maintenance of Heating, Ventilating, and Air Conditioning Systems in the United States Air Force: Final Report, School of Civil Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 18 November 1982.

16. Sanyo Basic Reference Manual. Users Manual. Sanyo Business Systems Corp., Computer Divisin, Moonachie NJ, undated.

17. Stoecker, W.F. ed. Procedures for Simulating the Performance of Components and Systems for Energy Calculations, (Third Edition). New York: ASHRAE Task Group, 1975.

18. Tom, Maj Steven T. "Maintainable Control Systems," ASHRAE Journal, 27 9: 38-40 (September 1985).

19. Seminar Abstract
Urbanczyk, James M. "A Manufacturer's Perspective", Direct Digital Control--Opportunities for Standardization. ASHRAE Transactions. 91 1B: x (1985).

20. Telephone Interview
Clark, Dr Donald R. US National Bureau of Standards. Personal Interview. US National Bureau of Standards, Gaithersburg MD, 3 February 1986.

21. Telephone Interview
Struthers, Larry. HQ AFESC/DEMM. Personal Interview. HQ AFESC, Tyndall AFB Florida, 29 August 1986.

22. Telephone Interview
Wilson, Edward E. HQ AFESC/DEMM. Personal Interview. HQ AFESC, Tyndall AFB Florida, 26 November 1985 and 29 August 1986.

VITA

Captain Steven J. Barlow was born on 21 February 1956 in Jersey City, New Jersey. He graduated from high school in North Arlington, New Jersey, in 1974 and attended the New Jersey Institute of Technology from which he received the degree of Bachelor of Science in Mechanical Engineering in May 1979. He received a commission in the USAF through the ROTC program in December, 1978. He was called to active duty in January, 1979 and served as design engineer and subsequently as the Chief, Contract Management at Shaw AFB, South Carolina. He then served as a Red Horse project engineer with the 554 CES(HR), at Osan AB, Korea from December, 1981 until December, 1982. Upon returning from Korea, he served as an industrial facilities engineer with the R&D, Civil Engineering Directorate, Aeronautical Systems Division at Wright-Patterson AFB. While serving at ASD, he received his registration as a Professional Engineer in the State of Ohio and served in this capacity until entering the Graduate Engineering Management Program of the School of Systems and Logistics, Air Force Institute of Technology in May, 1985.

Permanent address: 5 Foothill Drive
Kinnelon, New Jersey 07405

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A174617

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distritution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GEM/DET/86S-1	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Systems and Logistics 6b. OFFICE SYMBOL *(If applicable)* AFIT/DET	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS *(City, State and ZIP Code)* Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS *(City, State and ZIP Code)*
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS *(City, State and ZIP Code)*	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

11. TITLE *(Include Security Classification)*
See Box 19

12. PERSONAL AUTHOR(S)
Steven J. Barlow, B.S., P.E., Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Yr., Mo., Day)*	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	1986 September	145

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB. GR.	Computer Program, Control Systems, Heating, Ventilating, Air Conditioning
13	01		

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Title: PROTOTYPE OF A COMPUTER METHOD FOR DESIGNING AND ANALYZING HEATING, VENTILATING AND AIR CONDITIONING PROPORTIONAL, ELECTRONIC CONTROL SYSTEMS

Thesis Advisor: Steven T. Tom, Ph.D., Major, USAF
Assistant Professor of Mechanical Engineering

Approved for public release: IAW AFR 190-1.
[illegible]
LYNN E. WOLAVER 25 Sept 86
Dean for Research and Professional Development
Air Force Institute of Technology (AFC)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Steven T. Tom, Ph.D., Major, USAF	22b. TELEPHONE NUMBER *(Include Area Code)* 513-255-4552 22c. OFFICE SYMBOL AFIT/DET

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

The Air Force needs a better method of designing new and retrofit heating, ventilating and air conditioning (HVAC) control systems. Air Force engineers currently use manual design/predict/verify procedures taught at the Air Force Institute of Technology, School of Civil Engineering, HVAC Control Systems course. These existing manual procedures are iterative and time-consuming. The objectives of this research were to: (1) Locate and, if necessary, modify an existing computer-based method for designing and analyzing HVAC control systems that is compatible with the HVAC Control Systems manual procedures, or (2) Develop a new computer-based method of designing and analyzing HVAC control systems that is compatible with the existing manual procedures. Five existing computer packages were investigated in accordance with the first objective: MODSIM (for modular simulation), HVACSIM (for HVAC simulation), TRNSYS (for transient system simulation), BLAST (for building load and system thermodynamics) and Elite Building Energy Analysis Program. None were found to be compatible or adaptable to the existing manual procedures, and consequently, a prototype of a new computer method was developed in accordance with the second research objective. The prototype method developed the architecture needed to meet the manual procedure compatibility requirement and modeled three electronic components: a sensor, controller, and hi signal selector. The method incorporates four programs, written in BASIC, and copies of the programs, flowcharts, and sample runs are included. The method was developed to be easily expandable and recommendations for further development are given.

(Theses).